A METHOD FOR THE SIMULATION OF A HELICOPTER IN A TACTICAL ENVIRONMENT USING RAND'S TACTICS PROGRAM

By

Robert Alex Maier

United States
Naval Postgraduate School

THESIS

A METHOD FOR THE SIMULATION OF A HELICOPTER
IN A TACTICAL ENVIRONMENT
USING RAND'S TACTICS PROGRAM

by

Robert Alex Maier

Thesis Advisor: G. E. Heidorn

March 1971

Approved for public release; distribution unlimited.

A Method for the Simulation of a Helicopter in a Tactical Environment Using RAND's TACTICS Program

by

Robert Alex Maier
Lieutenant, United States Navy
B.S., United States Naval Academy, 1963

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1971

ABSTRACT

The TACTICS computer program developed by the RAND Corporation has the capability of simulating three-body, three-dimensional tactical situations. This study investigates the problem of using the TACTICS program when one of the bodies is to have the characteristics of a helicopter. The differences between the forces acting on a helicopter and the forces assumed by the TACTICS program are discussed, and a method to simulate the helicopter is presented. The basic idea of this method is to "fly" a hypothetical fixed-wing aircraft with parameters which enable it to perform like a helicopter. The programing required to simulate several basic helicopter maneuvers is discussed, and two examples of a helicopter in a tactical environment are presented.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

A three-body, three-dimensional simulation program entitled TACTICS was developed at the RAND Corporation primarily as an aid in the study of the close-in phase of two maneuvering fighter aircraft in a combat situation. The program was constructed in a modular form consisting of many subroutines, allowing the user to make modifications to the program for other areas of study. Reference 1 indicates possible applications of the program to the simulation of air-to-air, surface-to-air, and air-to-surface missiles, as well as ICBM reentry vehicle, rocket booster, and satellite investigation.

In recent years, the employment of the helicopter in conventional warfare has increased significantly. The missions of the helicopter now include not only the familiar search and rescue roles, but as mentioned in Ref. 2, profiles that cover the air-to-ground environment, including suppressive fire at area, light, and hard point targets. Helicopters have become fast and highly maneuverable, some having the capability of self-protection in hostile air and ground battle situations. While confrontations between helicopters and opposing force air-to-air and major surface-to-air defenses have been rare, incidents of this type may be expected to continue, and possibly increase, in future limited conflicts.

An analytical study of the helicopter in a tactical situation would attempt to provide answers to such diverse problems as optimum tactics to be employed by both the helicopter and opposing

weapon system, missile and sensor design parameters, and trade-offs between helicopter maneuverability and defensive armament. The TACTICS program has the capability of providing an output with vehicle position, velocity, acceleration, altitude, geometry, and aerodynamics listed by time history. However, in order to be able to provide accurate output, the program must be initialized with data that correctly describes the characteristics of the vehicle under consideration. These characteristic parameters have been developed quite thoroughly for certain high-speed vehicles, but the TACTICS manual, Ref. 1, does not indicate application of the program to studies of a vehicle with the unique capabilities of a helicopter.

B. OBJECTIVE

It is the objective of this study to devise a method for applying the TACTICS program to the problem of simulating a helicopter in a tactical environment, such as in an encounter with a high performance aircraft or when under attack from a ground installation.

C. ORGANIZATION OF THE PAPER

The paper is divided into seven sections. This section has presented the problem and the objective of the study. Section II describes the elements of the TACTICS program and discusses the forces and the generation of these forces in the program. Section III compares the forces on a helicopter with the forces in the TACTICS program, describes the helicopter to be simulated in this study, and discusses the parameters and the derivation of these parameters used in the simulation.

Basic maneuvers of the helicopter and the programing used to describe these maneuvers are presented in Section IV, along with several examples of the programing and computer output for the maneuvers. Section V presents two examples of the simulation of the hypothetical helicopter in a tactical situation, Section VI discusses modifications to the basic program, and the conclusions and recommendations are presented in Section VII.

II. THE TACTICS PROGRAM

The purpose of this section is to provide the reader with a general understanding of the concepts and theory of operation of the TACTICS program. This background will enable the potential user to comprehend more fully the difficulties involved in the simulation of a helicopter, and the reasoning behind the methods chosen. Detailed information concerning the program is available in the TACTICS manual, Ref. 1, but it should be noted that TACTICS was designed to allow the analyst to obtain information without an in-depth knowledge of the mechanics of the program. The TACTICS program can be considered to be composed of two elements, a group of computational routines and a Policy subroutine.

A. THE COMPUTATIONAL ROUTINES

The computational routines consist of the MAIN program and approximately fifty subroutines. The program reads the input data and initializes the starting positions and characteristics of three bodies, normally a fighter, a missile, and a target. As time is incremented in the program, these vehicles will "move" in a straight line in three-dimensional space according to the imposed initial conditions unless acted upon by some external forces. These forces, namely aerodynamic, gravitational, and thrust, are specified or generated by the program, and through the familiar relationship $F = ma$, an acceleration is determined. The routines then integrate the equations of motion to determine the velocity and position of the vehicles and provide an output of the details desired by the analyst.

The manner in which the forces act on the bodies is a result of the specifications in the Policy subroutine.

The computational routines are not easily modified to any great extent, but as noted earlier, this element of the TACTICS program does not have to be understood in detail to obtain useful results.

B. THE POLICY SUBROUTINE

The Policy subroutine is the means by which the analyst specifies the problem and dictates the desired performance of the vehicles under consideration. There are many guidance and control law routines available among the computational routines, and the Policy subroutine is used to state the criteria for changing from one routine to another. For example, if the requirement exists for a vehicle to fly straight and level for ten seconds, then start a 4.0 g climb, the sequence may easily be programmed in the Policy subroutine using a conditional GO TO statement based on time. Range and bearing angles are additional examples of criteria commonly used in the Policy subroutine. The applicable computational routines are listed in Appendix A, which includes a brief description of their function. When calling these subroutines in the Policy subroutine, the method in which the applied forces are to be developed is also specified. The development of these forces will be discussed in the following paragraphs.

C. DETERMINATION OF FORCES

1. Thrust (T)

There are three basic methods for determining the thrust force T on a body during any particular maneuver, the method

determined by the value of the variable ITHR.

a. ITHR = 1 or 2 causes a variable value of military or afterburner thrust to be applied to the body, the value determined as a function of mach number and altitude through the use of tables.

b. ITHR = 3 or 4 causes a constant military or afterburner thrust to be applied to the body, the value previously set in the initial data.

c. ITHR = 5 causes no thrust to be applied. This mode is normally used when simulating a point mass.

The value of thrust obtained by the above methods may be modified by the use of the variable THROTL. This parameter acts as a throttle setting and will adjust the thrust accordingly. The thrust force is assumed to act through the center of gravity of the body and along the longitudinal body axis.

2. Drag (D)

There are also three basic methods for determining the drag force D acting on a body, the method determined by the variable IAERO.

a. IAERO = 1 causes drag to be determined using analytic equations. The basic drag equations are:

$$D = C_D A q \; ; \; q = \tfrac{1}{2} \rho V^2$$

C_D = Drag coefficient
A = Aerodynamic reference area of the body - ft^2
q = Dynamic pressure - lb/ft^2
ρ = Air density - $slug/ft^3$
V = Magnitude of velocity of the body - ft/sec

C_D is determined by the following expression:

$$C_D = C_{D_O} + \frac{dC_D}{d(C_L^2)} C_L^2$$

C_{D_O} = Zero-lift drag coefficient

$\frac{dC_D}{d(C_L^2)}$ = A coefficient used with a parabolic function for drag coefficient

These coefficients are read in as initial data.

b. IAERO = 2 causes drag to be determined using the basic drag equations given above, but the value of the drag coefficient C_D is determined from tables as a function of mach number and lift coefficient.

c. IAERO = 3 causes no drag on the body. This mode is normally used in the simulation of a constant speed vehicle.

The drag force is assumed to act to directly oppose the velocity of the vehicle.

3. Lift (L)

The lift force acting on a body is also determined using analytic functions. The following expressions are used:

$$L = C_L A q$$

$$C_L = \frac{dC_L}{d\alpha} (\alpha - \alpha_o)$$

where C_L = Lift coefficient

$\frac{dC_L}{d\alpha}$ = Slope of the lift vs. angle of attack curve

α = Angle of attack

α_o = Zero-lift angle of attack

The lift force is assumed to act through the center of gravity normal to the plane containing the velocity vector and the pitch axis of the vehicle. The angle of attack (α) is defined to be the angle between the thrust vector and the velocity vector.

4. Gravitational force

In this problem, gravity is assumed to act normal to a flat earth, causing an acceleration of 32.174 ft/sec^2.

III. THE HELICOPTER

The TACTICS program was developed primarily for aerodynamic bodies having high speed capabilities. This section discusses the differences between the applied forces for these high-speed vehicles and the forces acting on a helicopter. A hypothetical helicopter is introduced to be used as an example, and the parameters used to describe it in the TACTICS program are presented.

A. DISCUSSION OF FORCES

Since the TACTICS program was designed for the simulation of vehicles such as a fighter aircraft or a missile, the forces used in the program are specified and generated in the conventional manner explained in the following paragraphs.

The thrust force acts along the positive longitudinal axis of the body to provide the desired acceleration along this axis. Conventional aircraft and missiles do not have the capability of changing the direction of the thrust in relation to the body. Deceleration of the vehicle along the longitudinal axis must be caused by a drag force greater than the thrust force, since the magnitude of this thrust cannot be made negative.

The lift force acting on a conventional body is generated by an aerodynamic surface moving through an air mass. This force is considered to act normal to the velocity vector of the body and causes a lateral or turning acceleration. An aerodynamic surface, such as a wing, having no relative velocity with respect to an air mass will not produce lift.

Drag is a force produced by the body as it moves through the air mass and generates lift. It is considered to act basically along the longitudinal axis to oppose the forward motion of the body. As in the case of lift, drag is not produced by a body at rest in the air mass.

The case of the forces acting on a helicopter during the hovering evolution, that is, with the fuselage at rest with respect to the air mass, is now considered. Lift is produced by the action of the rotor blades moving through the air, and by proper adjustment of blade speed and angle of attack, the lift force is made to balance the force due to gravity, resulting in a net acceleration of zero. Within the confines of the TACTICS program however, lift is a function of the square of the velocity. This implies that as the velocity of the body approaches zero, the coefficient of lift must become infinite to sustain the body in flight, an impossible condition.

Vertical climbs and descents are an extension of the hovering evolution. The lift force must be varied to produce a net vertical acceleration, but with a forward velocity of zero, this is not possible.

Investigation of force resolution in simple level flight produces similar problems. The helicopter may be considered to act as a body producing a single force vector. The component of the force vector along the flight path is labeled thrust, while the vertical component is considered to be the lift. Thrust and lift are therefore related in a complicated way through blade speed, angle of attack, orientation of the rotor disc, and several other variables. Conversely, thrust is basically independent of lift in the TACTICS program. Problems of this nature also arise in relating drag forces on a helicopter to the drag force in the TACTICS program.

Consideration was given to the translation of helicopter forces to the forces in the simulation program. Attempts to isolate the generation of lift by using blade area and rotor speed proved unsuccessful however, as did direct control of the direction of the force vectors. Reference 3 discusses the simulation of a helicopter rotor, but the method used was not considered practical for TACTICS applications.

In normal operation of the program, if the thrust force were set at a given value and a climb control-law routine were called, the TACTICS program would cause the vehicle to climb at the proper rate for the given thrust. Since it was not possible to resolve the helicopter forces and to determine the proper coefficients, the approach taken was to simulate the helicopter by specifying the performance parameters, rather than the forces. For example, this method specifies the proper rate of climb for the helicopter under consideration using published performance curves, then the appropriate forces to produce this rate of climb are developed. In effect, a fixed-wing vehicle is used having theoretical parameters that enable it to perform like a helicopter.

In general, the motion of the body is controlled by proper adjustment of the thrust force to obtain a desired acceleration, and by control of the velocity vector for the correct direction. Specific maneuvers and their operation are discussed in detail in Section V.

B. HELICOPTER CHARACTERISTICS

Reference 4 discusses methods for the performance analysis of an existing or contemplated helicopter. Reference 5 provides a

sample performance calculation for a theoretical helicopter using the guidelines of Ref. 4. While the performance capabilities of this helicopter are somewhat less than those of a modern helicopter, it is felt that the methods of analysis used are applicable to the study of the newer versions. For this reason, the hypothetical helicopter of Ref. 5 was chosen as the demonstration model for this simulation. The pertinent characteristics of this vehicle are shown below.

Specifications for the Hypothetical Helicopter

Type	Single-Rotor Cargo Utility
Rotor Diameter	52.0 feet
Gross Weight Range	5000-7000 pounds

A summary of the performance characteristics for a gross weight of 6300 pounds is given below:

Vertical rate of climb at sea level	750 fpm
Hovering ceiling (Out of ground effect)	5500 feet
Best climbing speed at sea level	55 mph
Best rate of climb at sea level	1450 fpm
Service ceiling	13,300 feet
Maximum level flight speed at sea level	118 mph
Average cruising speed	84 mph

Since the basic system of measurement in the TACTICS program is the foot-pound-second, some of the specified values must be converted to this system.

C. PARAMETER DEFINITION

This section contains a description of the parameters used in the TACTICS program that cause vehicle three to assume the characteristics of the hypothetical helicopter.

1. DATA 44: W0(3) = 6000.0

Initial weight was set at the average value of the gross weight range. Since fuel burning rate was not considered in the simulation, this value remained constant.

2. DATA 49: AREA(3) = 2123.7

The aerodynamic reference area was arbitrarily selected to be that of the rotor disc. Since the program assumes a given performance capability, rather than calculates it, this value is not critical. If the reference area is selected to be a very small value however, the lift coefficient must correspondingly become large to sustain the vehicle in flight. This can cause overflow to occur during execution of the program since the value of the lift coefficient is squared in several program statements.

3. DATA 54: ASMAX(3) = 3.0

Maximum lateral acceleration of the theoretical helicopter was estimated to be 3.0 g's.

4. DATA 90: CLMAX(3) = 10^6

The maximum coefficient of lift was assumed to be infinite (e.g., 10^6). This was necessary in order for vehicle three to develop the required lift at low airspeeds.

5. DATA 91: CD0CON(3) = .014

It is difficult to directly compare the drag forces produced by a helicopter and a fixed wing aircraft in the same reference frame.

A fixed wing aircraft at rest in the air mass obviously does not produce drag, nor does it fly. The rotor blades of a helicopter in a hover, however, produce profile drag and induced drag while generating lift. At higher speeds, both bodies are subjected to parasite drag, caused by the friction of the air mass on the fuselage. This element of total drag is the predominant force opposing motion of the vehicle in flight, and it is possible to directly translate the parasite drag of the hypothetical helicopter into the TACTICS reference frame.

Table VI of Ref. 5 is produced in part below:

V	RHP
59	12
88	39
117	88
147	168
176	286
205	444

where V is the velocity of the helicopter in feet per second, and RHP is the horsepower required to overcome parasite drag. To convert the power to actual drag in pounds, we use the equation

$$D = \frac{550\ RHP}{V} \quad .$$

Using the analytic functions, drag is determined by the expression

$$D = \tfrac{1}{2}\ C_D A \rho\ V^2 \quad .$$

Simultaneous solution of these equations yields

$$C_D = \frac{1100\ RHP}{\rho\ A\ V^3} \quad .$$

Using the values listed above for RHP and V, A equal to AREA(3), and an average value of ρ equal to .002, C_D is found to range only from .013 to .015. An average value of .014 was selected for this drag coefficient.

6. DATA 92: BCON(3) = 10^{-5}

This is a coefficient relating the change in the drag coefficient to the square of the lift coefficient. It may also be used to relate the increase in drag caused by the increase in angle of attack as a fixed wing body decreases in velocity. Since the induced drag is not considered for the helicopter, this coefficient should be zero. During program execution, however, this coefficient appears in the denominator of certain expressions, and was therefore set at a small number other than zero.

7. DATA 93: SLOPE(3) = 20,000

This is a coefficient relating the change in lift coefficient with a change in angle of attack. For a given airfoil, this is a definite value. Since the lift coefficient for the helicopter has a range that theoretically becomes infinite, the angle of attack must also become infinite to achieve this lift coefficient. Therefore, the large value of $\frac{dC_L}{d\alpha}$ was selected to prevent the angle of attack from becoming large, and has the effect of maintaining a level fuselage attitude even at very low velocities.

8. DATA 94: ALPHA0(3) = 0.0

Zero-lift angle of attack was arbitrarily set at zero, that is, level flight.

9. DATA 105: THCON(3) = 8,000.0

Since thrust must exceed the weight of a body in order for the body to accelerate in a vertical climb, constant military thrust was set at a value to provide 2000 lbs. excess thrust if desired. Since the acceleration is a limiting value in this simulation, and the thrust required to produce this acceleration is obtained through the use of the throttle parameter, this value is not critical.

IV. HELICOPTER MANEUVERS

This section describes the programing used to accomplish the simulation of the basic maneuvers of the hypothetical helicopter. It was the basic objective during the writing of these maneuver routines to test the feasibility of the method under consideration. Since it was extremely difficult to trace a given sequence of operations through the computational routines, it was generally not known in advance if a particular segment of programing would produce a desired result. During this debugging phase, the computational routines were compiled and placed in permanent storage in the computer system. All programing to test a given maneuver was done in the Policy subroutine, which was then compiled separately, linked to the computational routines, and executed. The use of the Policy subroutine in this manner was justified by the ease with which a programing segment could be modified, and by the lack of detail actually desired. An expanded and more detailed version of the helicopter program would probably use individual subroutines to accomplish a given maneuver, similar to the control-law subroutines incorporated for fighter aircraft.

A. THE HOVER

To preclude any difficulties in the generation of lift at zero airspeed, the helicopter is required to maintain a velocity greater than the arbitrarily set value of 0.5 ft/sec. It is felt that for all practical purposes, this velocity has no effect on a tactical situation, and the helicopter is essentially in a hover. Once this speed is

attained, it is possible to maintain the hover by balancing the gravitational force with an equal and opposite thrust force. Since the vehicle is essentially a motionless mass in three-dimensional space, however, it is treated as such. By calling the subroutine STRFLT(3, IAERO, ITHR), with IAERO = 3 and ITHR = 5, the vehicle will maintain a constant direction with no change in velocity.

It is of interest here to discuss the effects of certain subroutines when the vehicle is in a hovering condition described above or at a low velocity. Suppose the vehicle is in a hover with the velocity vector in a horizontal plane with a magnitude of 0.5 ft/sec. It is desired to have the vehicle climb at a 45 degree angle from the hover, requiring that the velocity vector be rotated 45 degrees above the horizontal. This can easily be accomplished by using the subroutine CLIMB1(I, GFORC, IAERO, ITHR), which is a constant-velocity, constant-g climb, but consideration must be given to the argument GFORC, the amount of g specified. A value of GFORC equal to 3.0 will almost instantly cause the velocity vector to exceed the 45 degree angle desired, since there is a large force (three times the body weight) applied to the body in the vertical direction. To prevent this large overshoot, GFORC should be set at a value near to 1.0. If diving and turning subroutines are used at low velocities, similar problems occur. Using values near unity permits the use of conditional IF statements in the Policy subroutine to monitor the velocity vector angle and prevents large discrepancies between desired and actual angles. The following discussion of vertical flight has an example that terminates with the hypothetical helicopter in a hover.

B. VERTICAL FLIGHT

To perform vertical maneuvers, the vehicle in the TACTICS program must be oriented with the thrust vector parallel or anti-parallel to the gravitational force vector. Using analytic functions, some lift will be generated as the body ascends or descends, causing a horizontal component of the resultant force vector acting on the body. This component will be small due to the low velocities associated with vertical flight, and can be neglected or considered a minor error in the pilot's ability to maintain perfectly vertical flight. Should the need arise for completely vertical flight, the velocity vector can be adjusted to provide a component of thrust to oppose the lift, but in this initial study, this accuracy was not justified.

Reference 5 calculates the vertical rate of climb at sea level for different gross weights. For the theoretical helicopter, this value is 1150 ft/min at 6000 pounds. Although calculations for the vertical rate of climb as a function of altitude were not performed, Ref. 5 indicates the general shape of the graph, and Figure 14-7 of Ref. 2 shows the curve for an actual helicopter. It may be noted that for gross weights in the mid range and altitudes below 7000-8000 feet, the rate of climb is essentially constant.

In the simulation, the variable VERCLB was set to 1150.0 and later transformed to feet per second. Since the vertical climb is used mainly for takeoffs, the constant assumption in this simulation is considered appropriate. For more advanced use, it is a relatively simple matter to make a table of VERCLB vs. altitude and thus determine the rate of climb appropriate for any given altitude. This technique is demonstrated in the section covering autorotation and forward climb.

Since the ability to increase the velocity in a vertical direction is dependent upon the thrust over and above that thrust required to counteract the body weight, it was decided to use vertical acceleration (caused by this excess thrust) as a parameter in determining the vehicle's vertical movement. Although this data was not available for the hypothetical helicopter, it is assumed that this data can be calculated or determined through flight tests, and an estimated value of 3.5 ft/sec^2 was used to initialize the variable VACCEL. Similarly, a variable VDECEL was initialized at 27.0 ft/sec^2 to indicate the maximum rate of acceleration for a vertically descending body. It was assumed that the helicopter would not be permitted to reach the free-fall acceleration of 32 ft/sec^2 for control restrictions.

While the variables described above were initialized in the Policy subroutine, it would be a relatively simple procedure to modify the computational routines to allow the accelerations and climb rates to be read in the same form as the standard input. Acceleration capabilities could also be related to altitude if this accuracy is desired.

Once these limits on vehicle performance have been established, it is necessary to determine the thrust required to execute the vertical maneuvers. The force required to accelerate a mass M with acceleration VACCEL is M * VACCEL. For the helicopter, M = WEIGHT(3)/G, and therefore the net force required for vertical acceleration is WEIGHT(3)*VACCEL/G. Neglecting lift, this net force is the resultant of the thrust, drag, and gravitational forces acting along the vertical flight path. Modification of equation (10) of Ref. 1 yields

$$WEIGHT(3)*VACCEL/G = T * COS\ \alpha - DRAG(3) - WEIGHT(3)*SIN\ \gamma .$$

Since T = THROTL(3)*THCON(3), the throttle setting necessary to cause an acceleration of VACCEL is determined by

$$THROTL(3) = \frac{\frac{WEIGHT(3)*VACCEL}{G} + DRAG(3)+WEIGHT(3)*SIN(V(3,6))}{THCON(3)*COS(ALPHA(3))} .$$

Similarly, the throttle setting for a constant rate of climb is determined by setting thrust equal to the drag and weight as follows:

$$THROTL(3) = \frac{WEIGHT(3)*SIN(V(3,6)) + DRAG(3)}{THCON(3)*COS(ALPHA(3))} .$$

In order to decelerate at a rate VDECEL, the throttle parameter must be decreased to the setting

$$THROTL(3) = \frac{\frac{-WEIGHT(3)*VDECEL}{G} + DRAG(3)+WEIGHT(3)*SIN(V(3,6))}{THCON(3)*COS(ALPHA(3))} .$$

The statements listed in Fig. 1 pertain to the hypothetical helicopter and have been extracted from a Policy subroutine. These statements will cause the helicopter to:

a) Perform a vertical takeoff after one second of elapsed time

b) Accelerate at VACCEL ft/sec^2 until maximum vertical climb rate of VERCLB ft/min is attained

c) Climb vertically to an altitude of 85.0 feet

d) Decelerate at VDECEL ft/sec^2 to a hover

Initial conditions established the vehicle at an altitude of one foot with a horizontal velocity of 0.5 ft/sec. Fighter and missile values were set at zero. A computer plot of the position of the helicopter in the vertical plane is shown in Fig. 2. The position was plotted at 0.5

```
C **** FIGHTER COMMANDS ***
C
      CALL CAPFLT(1,2)
C
C **** MISSILE COMMANDS ****
C
      CALL CAPFLT(2,2)
C
C **** HELICOPTER COMMANDS ****
C
      GO TO (310,320,330,340,350,360,370,380),LPOL
C     CONVERT CLIMB RATE TO FT/SEC
  310 VERCLB=VERCLB*0.01667
C     HOVER FOR ONE SECOND
  320 IF (TIME .GE. 1.0) GO TO 330
      CALL STRFLT(3,3,5)
      LPOL=2
      GO TO 390
C     CLIMB VERTICALLY
  330 IF (V(3,6) .GE. 89.9*RAD) GO TO 340
      CALL CLIMB1(3,1.1,3,5)
      LPOL=3
      GO TO 390
C     ACCELERATE TO PROPER RATE OF CLIMB
  340 IF (V(3,3) .GE. VERCLB) GO TO 350
      THROTL(3)=((WEIGHT(3)*VACCEL/G)+DRAG(3)+WEIGHT(3)*
     1SIN(V(3,6)))/(COS(ALPHA(3))*THCON(3))
      CALL STRFLT(3,1,4)
      LPOL=4
      GO TO 390
C     CLIMB AT PROPER RATE OF CLIMB
  350 IF (R(3,3) .GE.  85.0) GO TO 360
      THROTL(3)=(WEIGHT(3)*SIN(V(3,6))+DRAG(3))/(THCON(3)*
     1COS(ALPHA(3)))
      CALL STRFLT(3,1,4)
      LPOL=5
      GO TO 390
C     DECELERATE TO A HOVER AFTER PASSING 85 FT
  360 IF (V(3,6) .LE. 0.5) GO TO 370
      THROTL(3)=((-WEIGHT(3)*VDECEL/G)+DRAG(3)+WEIGHT(3)*
     1SIN(V(3,6)))/(COS(ALPHA(3))*THCON(3))
      CALL STRFLT(3,1,4)
      LPOL=6
      GO TO 390
C     LEVEL OFF AND HOVER
  370 IF (V(3,6) .LE. 0.0) GO TO 380
      CALL DIVE1(3,0.95,3,5)
      LPOL=7
      GO TO 390
  380 CALL STRFLT(3,3,5)
      LPOL=8
  390 CONTINUE
```

Figure 1. Extract from Policy subroutine with programing for vertical climb maneuvers.

Figure 2. Plot of vertical flight maneuvers in the Y-Z plane.

second intervals, with the unequal spacing of the points indicating accelerations.

C. HORIZONTAL FLIGHT

Horizontal flight is accomplished by a method analogous to that of vertical flight. In this case, a horizontal acceleration and deceleration capability of HACCEL = 8.0 ft/sec^2 and HDECEL = 5.0 ft/sec^2 is assumed. The vehicle is oriented with the velocity vector in the horizontal plane and the throttle is adjusted to produce the thrust required for the desired acceleration. In this case, drag becomes more noticeable due to the higher velocities attained, and acceleration will be somewhat less than in the absence of drag, while deceleration will be correspondingly higher. The thrust force in this case is assumed to be a constant, dependent on the position of the rotor disc. Since the rotor disc may be tilted backward, the net force produced by the disc may have a component directly opposing forward flight. This condition does not exist with fixed wing aircraft, but it can be simulated in the program by the use of a negative throttle parameter.

A helicopter in forward flight has many of the characteristics of a conventional aircraft, and can be simulated in the TACTICS program using the normal analytic functions. For example, a diving turn can be accomplished by lowering the velocity vector below the horizontal using a DIVE subroutine, then calling a TURN subroutine to complete the maneuver. As before, acceleration along the flight path must be controlled by the thrust, unless constant-speed routines are used. Excess thrust during a diving turn will cause the vehicle

to accelerate, and if the turn is specified as constant-g, the flight path will describe an increasing spiral.

Use of a TURN subroutine when the vehicle is at a very low velocity has the effect of pivoting the vehicle about the vertical axis. A helicopter can easily accomplish this maneuver, and it may prove useful in providing a stable platform from which an opponent can be continuously tracked.

The extract from a Policy subroutine listed in Fig. 3 shows the statements that will cause the helicopter to:

a) Decelerate to a hover at HDECEL ft/sec^2

b) Hover for ten seconds

c) Accelerate at HACCEL ft/sec^2

d) Start a diving right turn of 1.2 g's after attaining a velocity of 88.0 ft/sec (60 mph)

Initial conditions located the helicopter at 4000 feet with a velocity of 60.0 ft/sec. A computer plot of this maneuver in the horizontal plane is shown in Fig. 4. The apparent left turn of the vehicle is due to the orientation of the axes in the TACTICS program, as explained in Ref. 1. As before, the uneven spacing of the position points plotted at one second intervals indicates accelerations of the vehicle.

D. FORWARD FLIGHT CLIMB

For a given gross weight and optimum climb speed, the rate of climb of a helicopter will decrease with altitude. The service ceiling of the helicopter is determined by that altitude where the rate of climb is less than 100 ft/min. This section demonstrates a method that includes a table of rate of climb vs. altitude to simulate a realistic forward climb flight path. This method may be applied to other

```
C **** FIGHTER COMMANDS ****
C
      CALL CAPFLT(1,2)
C
C **** MISSILE COMMANDS ****
C
      CALL CAPFLT(2,2)
C
C **** HELICOPTER COMMANDS ****
C
      GO TO (310,320,330,340,350,360),LPOL
C     DETERMINE FORCE REQUIRED FOR DECELERATION
  310 THFOR=WEIGHT(3)*HDECEL/G
C     ADJUST THE THROTTLE FOR THIS FORCE
      THROTL(3)=-(THFOR/THCON(3))
C     SLOW TO 0.5 FT/SEC
  320 IF (V(3,4) .LE. 0.5) GO TO 325
      CALL STRFLT(3,1,4)
      LPOL=2
      GO TO 390
  325 HTIME=TIME
C     HOVER FOR 10.0 SECONDS
  330 IF (TIME-HTIME .GE. 10.0) GO TO 335
      CALL STRFLT(3,3,5)
      LPOL=3
      GO TO 390
C     DETERMINE FORCE REQUIRED FOR ACCELERATION
  335 THFOR=WEIGHT(3)*HACCEL/G
C     ADJUST THE THROTTLE FOR THIS FORCE
      THROTL(3)=THFOR/THCON(3)
C     ACCELERATE TO 60.0 MPH
  340 IF (V(3,4) .GE. 88.0) GO TO 350
      CALL STRFLT(3,1,4)
      LPOL=4
      GO TO 390
C     LOWER THE NOSE 15.0 DEGREES BELOW THE HORIZON
  350 IF (V(3,6) .LE.-15.0*RAD) GO TO 360
      CALL DIVE1(3,0.9,3,5)
      LPOL=5
      GO TO 390
C     START A 1.2-G RIGHT TURN
  360 CALL RTRN1(3,1.2,3,5)
      LPOL=6
  390 CONTINUE
```

Figure 3. Extract from Policy subroutine with programing for horizontal maneuvers

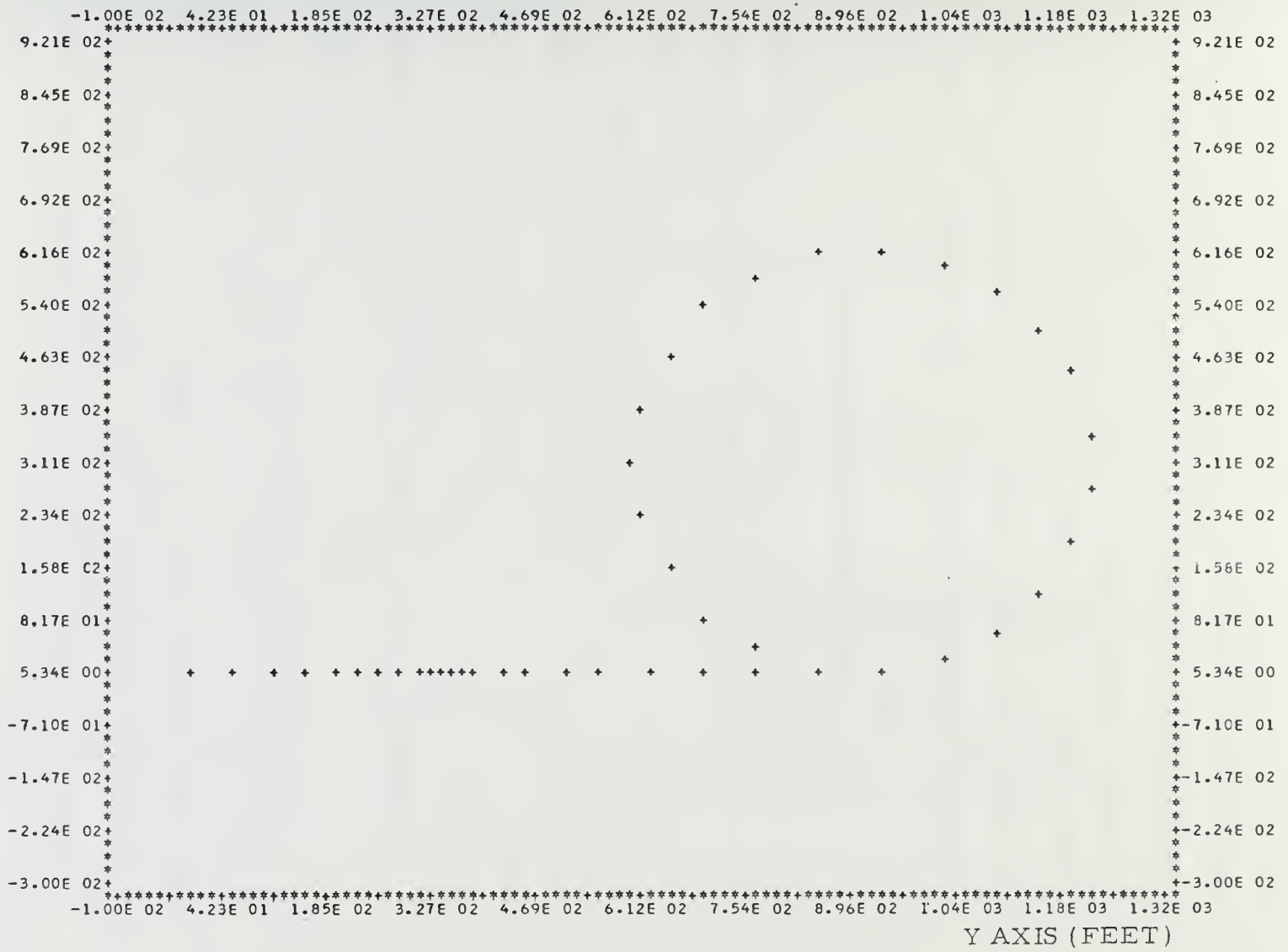


Figure 4. Plot of horizontal flight maneuvers in the X-Y plane.

maneuvers in the simulation, such as the vertical rate of climb vs. altitude problem mentioned earlier.

Figure 19 of Ref. 5 illustrates forward rate-of-climb capability vs. altitude at normal rated power for the hypothetical helicopter. This graph was used to initilize an array named FWDROC with paired values of rate of climb and altitude. During program execution, the array is scanned to find the proper rate of climb for the current altitude of the helicopter. The velocity vector of the vehicle is then adjusted by CLIMB or DIVE subroutines to make the vertical component of this vector equal to the proper rate of climb. Since there is normally some overshoot due to the integration step size when adjusting to the desired climb angle, an error of up to 24.0 ft/min is allowed in order to prevent constant climbing and diving about the correct value.

Optimum forward velocity for the best rate of climb is determined by constructing a horizontal tangent to the power-required curve for the hypothetical helicopter, Fig. 10 of Ref. 5. The best climb speed determined by this technique is 55 mph. The helicopter is normally accelerated to this velocity in level flight prior to initiating the climb.

An example of this climb method is illustrated in Fig. 5. The initial position of the helicopter is at 5000 feet of altitude with a velocity of 40.0 ft/sec. Figure 6 is a plot of the flight path of the vehicle in the vertical plane, and the decreasing slope of the curve indicating a decreasing rate of climb may be noted. The higher altitudes were chosen for the example, since the climb rate is almost constant in the lower ranges.

```
C **** FIGHTER COMMANDS ****
C
      CALL CAPFLT(1,2)
C
C **** MISSILE COMMANDS ****
C
      CALL CAPFLT(2,2)
C
C **** HELICOPTER COMMANDS ****
C
      GO TO (310,320,330,340,350),LPCL
C     CONVERT INITIAL DATA TO PROPER UNITS
  310 DO 20 I=1,10
   20 FWDROC(I,2)=FWDROC(I,2)*0.01667
      CLBOPT=CLBOPT*1.467
C     ACCELERATE TO OPTIMUM CLIMB SPEED
  320 IF (V(3,4) .GE.CLBOPT) GO TO 330
      THROTL(3)=((WEIGHT(3)*HACCEL/G)+DRAG(3)+WEIGHT(3)*
     1SIN(V(3,6)))/(THCON(3)*COS(ALPHA(3)))
      CALL STRFLT(3,1,4)
      LPOL=2
      GO TO 390
  330 IF (R(3,3) .GE. 12500.0) GO TO 340
C     DETERMINE CORRECT RATE OF CLIMB FOR ALTITUDE
      DO 30 I=1,10
      IF (R(3,3) .GE. FWDROC(I,1))VERCOM=FWDROC(I,2)
   30 CONTINUE
      THROTL(3)=(DRAG(3)+WEIGHT(3)*SIN(V(3,6)))/(THCON(3)*
     1COS(ALPHA(3)))
C     CHECK FOR CORRECT RATE OF CLIMB
      IF (V(3,3) .LT. (VERCOM-0.4)) GO TO 335
      IF (V(3,3) .GT. (VERCOM+0.4)) GO TO 336
      CALL STRFLT(3,1,4)
      LPOL=3
      GO TO 390
C     ADJUST CLIMB ANGLE FOR CORRECT RATE OF CLIMB
  335 CALL CLIMB1(3,1.1,1,4)
      LPOL=3
      GO TO 390
  336 CALL DIVE1(3,0.95,1,4)
      LPOL=3
      GO TO 390
C     LEVEL OFF AT SERVICE CEILING
  340 IF (V(3,6) .LE. 0.0) GO TO 350
      CALL DIVE1(3,0.95,3,5)
      LPOL=4
      GO TO 390
  350 CALL STRFLT(3,3,5)
      LPOL=5
  390 CONTINUE
```

Figure 5. Extract from Policy subroutine with programing for forward flight climb maneuver.

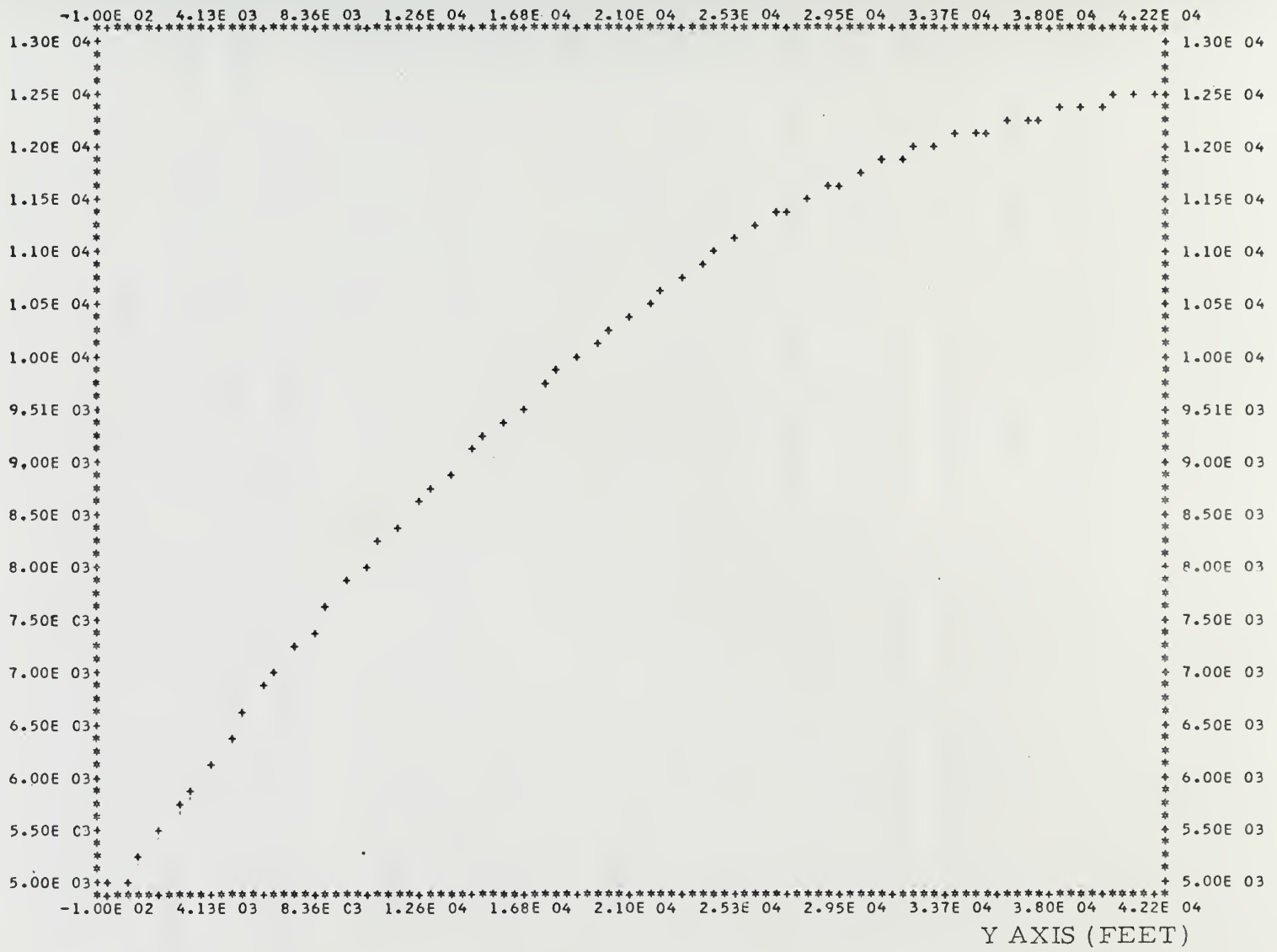


Figure 6. Plot of forward climb flight maneuver in the Y-Z plane.

This method is considered adequate for the purposes of this paper. For a more detailed analysis, interpolation routines should be used to provide a functional relationship between rate of climb and altitude, and the process would more appropriately be located in a subroutine.

E. FORWARD FLIGHT AUTOROTATION

Autorotation of the hypothetical helicopter is simulated by a method analogous to that employed in the forward climb maneuver, but in this case, the rate of descent is a function of forward speed. Figure 15 of Ref. 5 was used to initialize an array named AUTORO with paired values of rate of descent and forward speed. This array was then scanned to determine the rate of descent corresponding to the actual horizontal component of velocity of the helicopter. The angle of descent is then adjusted by CLIMB and DIVE subroutines, and the thrust regulated to cause the vehicle to follow the correct flight path for an autorotating helicopter. A more realistic approach to the programing of the method during an analysis would be to use subroutines and interpolation routines for a more accurate approximation of the flight path.

An example of the programing used to simulate the autorotation maneuver is used in the Policy subroutine demonstrating an autorotating helicopter under attack by a ground-to-air missile, Appendix B, Section III. A computer plot of the maneuver is given in Section IV of the same Appendix, but the scaling factor causes the maneuver to appear steeper than it actually is.

V. TACTICAL EXAMPLES

Two examples indicating potential applications for the methods developed in the simulation of a helicopter are presented in this section. The examples illustrate Policy subroutines that use combinations of the maneuvers described earlier, but no attempt was made to determine optimum maneuvering for either vehicle. An analysis of a tactical situation would normally consider only certain aspects of a given attack. Later developments would be considered separately. The fighter vs. helicopter example is extended beyond the initial attack by the fighter for demonstration purposes only.

A. FIGHTER VS. HELICOPTER

In this example, a fighter aircraft with an altitude advantage makes a diving gunnery run on the hypothetical helicopter, then pulls up and circles around for another attack with 20-mm projectiles. The fighter aircraft simulated is an F - 104, and tables are used to describe the interactions among the lift coefficient, drag coefficient, angle of attack, maximum coefficient of lift, and mach number. The Policy subroutine defines the problem as follows:

VEHICLE 1(F - 104 pursuing aircraft)

- Fly pursuit course navigation against helicopter; throttle set at .9.
- When relative range to helicopter is less than 4500 ft., fire two 20-mm projectiles at the target.
- If altitude falls below 3300 ft., climb at 5.5 g until nose is 3 degrees above the horizon; throttle at full military power.

- Perform a 4.5 g climbing right turn until bearing to the helicopter is less than 60 degrees, then fly pursuit course navigation with throttle set at .80.

- When relative range to helicopter is less than 4500 ft., fire two 20-mm projectiles and climb out at 5.5 g when altitude falls below 1000 ft.

VEHICLE 2 (20-mm projectile)

- Fly captive flight until launch criteria are satisfied.

- Launch and fly in a ballistic trajectory.

- Find miss distance, restore conditions to launch time, launch second projectile 0.01 seconds later.

VEHICLE 3 (Hypothetical helicopter)

- Perform constant 1.3-g turn toward the fighter to attempt to force a head-on passing situation.

- When fighter passes and starts to turn for re-attack, slow to 25 mph if possible.

- When fighter begins second tracking run, autorotate at current forward speed.

The initial position of the fighter is at the origin at an altitude of 7000 ft., with a heading of 090 degrees at 800 ft/sec. The helicopter is located 12,000 ft. directly East at an altitude of 3000 ft., heading 350 degrees at 120 mph. The complete Policy subroutine for this tactical problem is listed in Appendix B, Section I. Three orthogonal views of the flight paths of the fighter and the helicopter are shown in Section II of Appendix B. The points were plotted at two second intervals by the computer, and the connecting line segments were then added to make visualization of the flight paths easier. The turning helicopter causes the fighter to make constant corrections during the initial attack, and since the fighter is using pure pursuit techniques, rather than adding some type of lead calculation, the miss distance of the 20-mm projectile is approximately

100 feet. Similar results are obtained during the second attack, when the helicopter is autorotating at approximately 40 mph. The attacking figher was in a position for re-attack before the helicopter could slow all the way to 25 mph.

B. MISSILE VS. HELICOPTER

This example demonstrates the capability of the TACTICS program to simulate the helicopter in a ground-to-air missile environment. Due to the high velocity of the missile and its proportional navigation system, the helicopter attempts to evade the threat by autorotating at high speed when the relative range is less than 2.5 miles. The helicopter is initially 25,000 ft. away from the missile site at 1000 ft. of altitude. It is heading directly at the site at a speed of 120 mph, and at missile firing, speed is increased as much as possible within a 140 mph maximum prior to commencing the autorotation. The ground-to-air missile has the characteristics of the vehicle described in Example 3, Ref. 1. The Policy subroutine defines the problem as follows:

VEHICLE 1 (Not used)

VEHICLE 2 (Surface-to-air missile)

- Launch at time zero; thrust constant at 5397 lbs; initial velocity of 100 ft/sec.
- Fly in 0-g dive until burnout at 1.0 seconds.
- Continue 0-g until time to guide; no thrust.
- Begin guidance 1.0 sec. after launch; fly proportional navigation with time lag of 0.2 sec; no thrust.

VEHICLE 3 (Hypothetical helicopter)

- Fly straight and level at missile site until missile launch.

- Accelerate at maximum capability until range to missile is 2.5 miles.

- Autorotate at current forward speed.

The Policy subroutine for this scenario is given in Appendix B, Section III. The computer plot of the flight paths of the two vehicles in the vertical plane is shown in Section IV of Appendix B. The helicopter was able to increase its velocity only four mph prior to initiating autorotation. As expected, the miss distance of the missile was only 1.5 feet, due to the slow speed of the helicopter and the proportional navigation system used by the missile.

VI. PROGRAM MODIFICATIONS

Minor modifications to the TACTICS MAIN program were made to facilitate input/output operations on the computer. Reference 6 discusses operation of the TACTICS program at the Naval Postgraduate School, and was used as a reference for program execution during this study. A subroutine used at the Postgraduate School was also modified and added to the computational routines to add a plotting capability to the program. These modifications will be discussed in this section.

A. MAIN PROGRAM

The IBM 360/67 computer system at the Naval Postgraduate School was utilized for the execution of the program. The system normally operated under two modes, the TSS/360 time sharing system and the basic OS/360 system. Since the input and output statements differed in each mode, the variables INGAGE and OTGAGE were used in the READ and WRITE statements of the MAIN program to facilitate the change from one mode of operation to the other. These variables were also used in each subroutine having READ and WRITE statements, and were placed in COMMON/GAGE/. Under TSS/360 operation, the MAIN program asks for the values of INGAGE and OTGAGE, and the operator enters these values at the remote terminal. Under OS/360 operation, the desired values are entered as the first data card.

The main program was also modified by the addition of statements calling the subroutine OSPLOT at the termination of vehicle maneuvering. The arrays used as calling arguments in OSPLOT

were placed in COMMON/PLOTT/ and initialized to the value 900, 000. 0 at the beginning of the MAIN program. An explanation of these arrays and the function of OSPLOT is given in the following paragraphs. The modified MAIN program is listed in Appendix C, Section I.

B. SUBROUTINE OSPLOT

OSPLOT is a subroutine that was written at the Naval Postgraduate School for the purpose of plotting sets of points on a single graph. It was decided to use this subroutine to provide the user with a rough plot of the flight path of the vehicles under consideration, and to permit visualization of the maneuvers without scanning pages of computer output. Since the variable DTPO determines the time interval between information outputs, it was a logical choice for the time spacing of points on the flight path plot. During program execution, the Policy subroutine stores the coordinates of the vehicles, at the chosen points in time, in the arrays of COMMON/PLOTT/. This information is then used by OSPLOT at the completion of maneuvering to produce a two-dimensional plot of the flight path. The subroutine is normally called three times in the MAIN program with the arguments arranged to produce three orthogonal views of the maneuvers.

During the experimentation phase of maneuver testing, the automatic scaling features of OSPLOT were used for ease of operation. When further refinement of the plot was desired, manual scaling or adjustment of DTPO was employed. The printing of the axes was eliminated from the subroutine to make the output neater.

The subroutine was also modified to print different characters for each vehicle. OSPLOT requires that if the abscissa and the

ordinate of one vehicle are known and are to be plotted, the ordinate corresponding to the same abscissa for the other vehicle must also be known. Since the vehicles normally have no points in common, the value 900, 000. 0 (initialized in the MAIN program) was given as the ordinate of the second vehicle. When the program scans the coordinates for plotting, values greater than 800, 000. 0 are ignored, and the ficticious points are not plotted. The revised version of OSPLOT is listed in Appendix C, Section II.

VII. CONCLUSIONS AND RECOMMENDATIONS

The primary difficulty in applying the TACTICS program to the simulation of a helicopter in a tactical environment is the translation of the forces produced by the rotating blades of the helicopter to the conventional fixed-wing forces used in the TACTICS simulation program. After a study of this problem, it was concluded that if a transformation of the forces was even possible, it would require major new subroutines to be added to the existing computational routines. It was therefore decided to use the known capabilities of the helicopter as limiting values, and to use the computational routines already available to cause the vehicle to perform within these limits. This method utilizes the capabilities of the TACTICS program and allows helicopter and fixed-wing vehicles to be "flown" in the same simulation.

Additional work remains to be done in the programing of helicopter maneuvers, and the flexibility of the program in the simulation of the helicopter can be improved. The use of excess thrust in place of acceleration as a limiting factor is another area requiring additional investigation. Nevertheless, the ideas presented in this paper are feasible and represent one approach to the problem of simulating a helicopter in a tactical environment using the TACTICS program.

APPENDIX A

TACTICS SUBROUTINE DESCRIPTIONS

Subroutine	Description
MAIN	Zeros values in COMMON/TABLE/ Determines number of cases in run Determines location of input and output files; times run
INCOND	Prints initial data values Sets initial flags and zeros out variables Computes relative range rates and velocity vectors
OUTPUT	Prints output as controlled by NPRINT options specified in POLICY
AUXCOM	Computes auxiliary computations for printing only
TABLER	Reads tables when provided for in vehicle's aerodynamic characteristic description
DECRD	Reads data cards for information as provided on input data form
ATTITUD	Computes attitude angles
CLIMB1	Elementary climb where G-force is specified
CLIMB2	Constant speed climb ; depends on excess thrust
DIVE1	Simple dive, G-force is specified
STRFLT	Vehicle flies in a straight line but not necessarily level
STRLVL	Vehicle flies straight and at a constant altitude
RTRN1	Simple right turn; G-force specified determines angle of bank
RTRN2	Constant-altitude, constant-Mach right turn; depends on excess thrust

Subroutine	Description
RTRN3	Constant-Mach right turn where G-force is specified
LTRN1	Simple left turn; G-force specified determines angle of bank
LTRN2	Constant-altitude, constant-Mach left turn; depends on excess thrust
LTRN3	Constant-Mach left turn where G-force is specified
AIM	Aims vehicle one as specified in initial conditions
TABINT	Interpolates values as provided in input data tables
INTGRT	Computes relative and absolute positions, velocities, and ranges
LIMIT	Computes aerodynamic limitations as specified in the input data
CAPFLT	Zeros out variables for vehicle(s) not used or when attached to another vehicle, such as a missile
ATMOS	Computes model atmosphere, speed of sound and Mach number for given altitude
WEIGH	Computes weight of vehicle as fuel is consumed
THRUST	Computes thrust available either from data tables or an input constant
LAG	Computes change from commanded lateral acceleration (GFORC) to actual output acceleration
AERODN	Computes maneuver-induced drag, input acceleration components, and changes in velocity due to maneuvers
INITS	Initializes conditions for ARKAM integration
PLACRD	Computes throttle setting based on aircraft limitations, Policy input, and atmospheric conditions

Subroutine	Description
STORE	Stores position and velocity when using restore option
FUNCTN	Computes unit vector components of commanded acceleration and thrust
MAG	Computes magnitude of lateral commanded acceleration
COORD	Makes transformations from rectangular to spherical coordinates and vice-versa
DAUX	Integrates cartesian and geocentric coordinates for all vehicles relative to point of intercept
ARKAM	Integration subroutine
MACHR8	Calculates aerodynamics in reverse for constant-Mach maneuvers
TOPCON	Computes local and inertial coordinate transformations
DOT	Computes vector dot products
RATES	Computes range and angular rates of change
CROSS	Computes vector cross products
B20MM	Computes ballistic trajectory of 20-mm projectile
PRONAV	Proportional navigation subroutine for intercept problem
PRSUIT	Pursuit course navigation subroutine for intercept problem
LAUNCH	Subroutine called for the launching of a missile or projectile from vehicle
OSPLOT	Subroutine that outputs two dimensional plot of vehicle positions

APPENDIX B

I. COMPLETE POLICY SUBROUTINE FOR FIGHTER VS. HELICOPTER

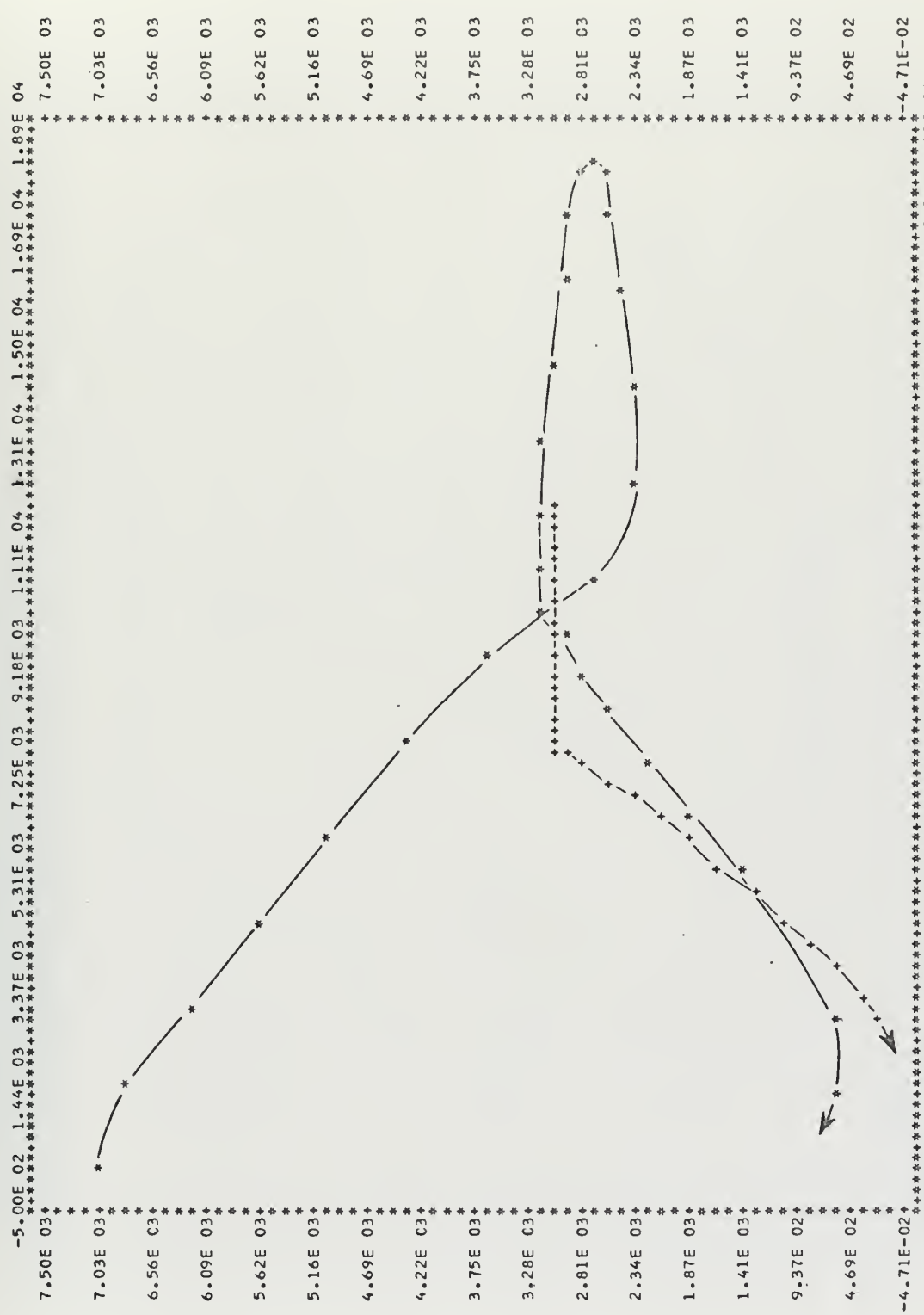
```
      SUBROUTINE POLICY
C
C
      REAL MACHCL,MACHCD,MACHT1,MACHT2,MACHTA,MACHLM,MACHMX
      REAL LAMCAO,LAMBDA,LIFT,MACH,IMPLSE,KLAUN,MINMR
      REAL LAT,LONG,LATO,LONGO
      COMMON/CATMOS/ TEMP(3),PRES(3),DENS(3),SOUND(3),MACH(3),Q(3)
      COMMON/VECTOR/ R(3,6),RREL(6,6),V(3,6),VREL(6,6)
      COMMON/AERO/   THRUST(3),ALPHA(3),CODRAG(3),COLIFT(3),DRAG(3),
     1               LIFT(3),WEIGHT(3),AREA(3),ASMAX(3),CLMAX(3),WO(3),
     2               CWDOT(3),THROTL(3),BETA(3)
      COMMON/RATE/   OMEGA(3,4),OMEGAR(6,4),OMEGAB(6,4),RDOT(6),VDOT(3),
     1               THDOT(3),PHIDOT(3)
      COMMON/ATT/    AZMUTH(6),ELEV(6),ROLL(3),ROLLRB(3),BEARNG(6)
     1               ,ELEVMX(3),AZMAX(3),BANK(3)
      COMMON/ACCEL/  ACOM(3),ACOMA(3),ACOMD(3),AOUT(3),AOUTA(3),AOUTD(3)
     1               ,GFORCE(3),ACLMAX(3),AMAX(3),F(3)
      COMMON/UNIT/   UNITV(3,3),UNITPV(3,3),UNITR(6,3),UNITL(3,3),
     1               UNITT(3,3),A(3,3),D(3,3),A1(3,3),D1(3,3)
      COMMON/INTEG/  HDP,T,VAR(18),DER(18),TEMPS(9,18),VARSTR(18),ICNT,
     1               STIME,TIME,DTO,HMIN,HMX,HM,ERTEST,DTMIN,DT,SHDP,NH
      COMMON/FLAG/   JPOL,KPOL,LPOL,MPOL,NPOL,JINTEG,ISTART,ILAUN,JATMOS
     1               ,ISTORE,IMISS,KINTEG,ICAP
      COMMON/PRINT/  IPRINT(20),TPO,TOTAL,DTPO,NPRINT,TITLE(18)
      COMMON/MISSL/  TGUIDE(3),TLAUN(3),TBURN1,TBURN2,KLAUN,RMTMAX,MINMR
     1               ,ABOOST(3),IMPLSE(3),WBURN(3)
      COMMON/ANALYT/ CDO(3),B(3),ALPHAO(3),CDOCON(3),BCON(3),THCON(3),
     1               TABCON(3),SLOPE(3)
      COMMON/CONST/  G,RAD,ITAU(3),TAU(3,5),DATA(200)
      COMMON/NAVIG/  SGAMA(3),LAMDAO(3),LAMDA(3),DVTH(3),DVPHI(3)
      COMMON/TABLE/  JVEH(3),MACHCL(3,42),CLMAXT(3,1,42),MACHCD(3,16),
     1               CLTCD(3,32),CDT(3,16,32),MACHT1(3,16),ALT1(3,16),
     2               THT1(3,16,16),MACHT2(3,16),ALT2(3,16),THT2(3,16,16)
     3               ,CLTA(3,13),MACHTA(3,22),ALPHAT(3,22,13),WDOT(3),
     4               WDOT1(3,16,16),WDOT2(3,16,16),ALTM(3,14),MACHMX(3),
     5               MACHLM(3,14)
      COMMON/EXTRA/  EXTR(20),IEXTRA(10)
      COMMON/VEH/    ACTION(3,2),DASMX(3,5),DUMCL(3,5),ACTNO(3)
```

```
      COMMON/GEOCNT/ RO(3),RODOT(3),RRDOT(3,6),UNITRR(3,3),UNITPP(3,3),
     1               UNITLL(3,3),FORCE(3,3),IROT8,RR(3,6),VI(3,6),
     2               RI(3,6),OMEGAE,LATO,LONGO,ALT(3),LAT(3),LONG(3),
     3               GRANGE(3)
      COMMON/SAVREL/ SAVMIS(1,6)
      COMMON/PLAC/   IPLAC(3),PLACC(3,3)
      COMMON/PLOTT/    XXXX(122),XXX(122,2),YYY(122),ZZZ(122,2)
      DOUBLE PRECISION T,HDP,SHDP
      DIMENSION AUTORO(8,2)
      DIMENSION LEVEL (3)
      DIMENSION FWDROC(7,2)
      DATA VERCLB/1150.0/
      DATA LEVEL/3*0/,TEST/0.0/,KK1/1/
      DATA VACCEL/3.5/,VDECEL/27.0/,HACCEL/8.0/,HDECEL/5.0/
      DATA CLBOPT/55.0/
      DATA FWDROC/0.0,3000.0,6000.,7000.,8000.,9000.,10000.,
     11600.,1550.,1500.,1350.,1200.,1050.,900./
      DATA AUTORO/0.0,20.0,40.0,60.0,80.0,100.0,120.0,140.0,
     12430.0,2050.0,1590.0,1462.0,1650.0,2092.,2815.,3800./
C
C *** SPECIFY OUTPUT ***
      NPRINT=3
      IPRINT(1)=1
      IPRINT(2)=2
      IPRINT(3)=3
C
C *** FIGHTER COMMANDS ***
C
      GO TO (110,120,140,150),JPOL
 110  CONTINUE
      II=0
      IMISS=1
      ISTORE=1
 120  CONTINUE
      IF (R(1,3) .LE. 3300.0) GO TO 130
      THROTL(1)=.9
      CALL PRSUIT(1,2,2)
      JPOL=2
      GO TO 190
 130  CONTINUE
      CALL CLIMB1(1,5.5,2,1)
      IF(V(1,6) .GE. 3.0*RAD) JPOL=3
      IF (V(1,6) .GE. 0.0*RAD) LPOL=3
      GO TO 190
 140  IF (ABS(BEARNG(2)) .LE. 60.0*RAD) GO TO 145
      CALL RTRN1(1,4.5,2,2)
      GO TO 190
 145  KPOL=1
```

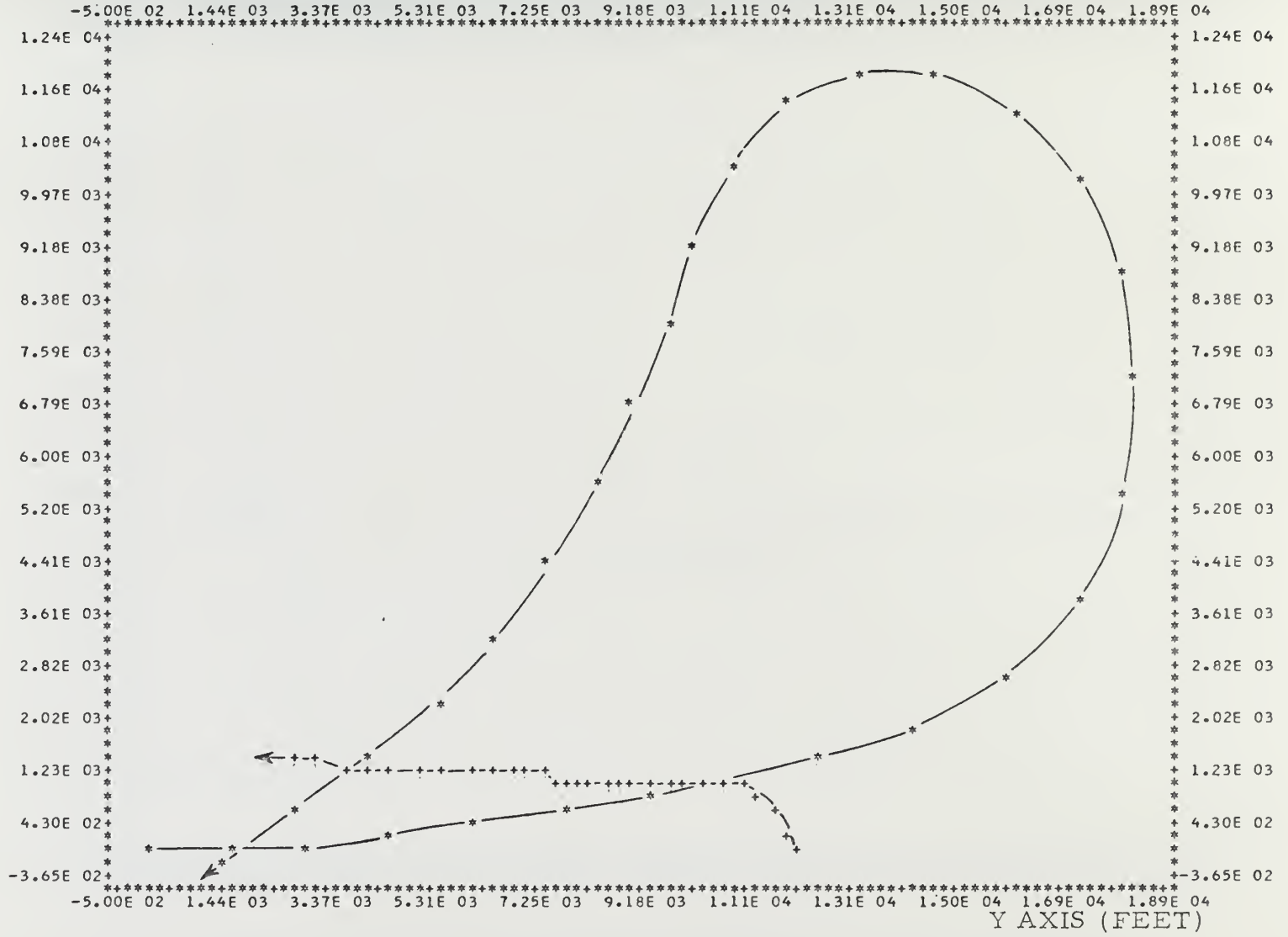
```
      LPOL=6
  150 JPOL=4
      THROTL(1)=0.8
      IF (R(1,3) .LE. 300.0) GO TO 130
      IF (RREL(2,4) .LE. 1000.0) GO TO 130
      CALL PRSUIT(1,2,2)
  190 CONTINUE
C
C *** MISSILE COMMANDS ***
C
      GO TO (210,220,230,240,250,215),KPOL
  210 CONTINUE
      II=0
      IF (RREL(2,4) .LE. 4500.0 .AND. ABS(BEARNG(2)) .LE. 30.*RAD) GO TO
     1 220
  215 CALL CAPFLT(2,1)
      JINTEG=3
      GO TO 290
  220 CONTINUE
      CALL LAUNCH(2,2800.0,3,5)
  230 CONTINUE
      IF (ISTORE .EQ. 0) GO TO 240
      CALL B20MM(2)
      KPOL=3
      GO TO 290
  240 CONTINUE
      JINTEG=2
      IMISS=1
      ISTORE=1
      KPOL=5
      IF (II .EQ. 1) KPOL=6
      KK=0
  250 CONTINUE
      IF (KK .NE. 0) GO TO 220
      CALL CAPFLT(2,1)
      KK=1
      II=II+1
      GO TO 290
  290 CONTINUE
C
C **** HELICOPTER COMMANDS ****
C
      GO TO (310,320,330,340,350,360,370),LPOL
  310 CONTINUE
      IF (V(3,5) .LT. -90.0*RAD) GO TO 320
C     TURN TO OPPOSITE HEADING OF FIGHTER
      CALL RTRN1(3,1.3,3,5)
      GO TO 390
```

```
  320 CALL STRFLT(3,3,5)
      LPOL=2
      GO TO 390
C     SLOW TO 25 MPH IF POSSIBLE
  330 THFOR=WEIGHT(3)*HDECEL/G
      THROTL(3)=-(THFOR/THCON(3))
      LPOL=4
  340 IF (V(3,4) .LE. 36.6)  GO TO 350
      CALL STRFLT(3,1,4)
      LPOL=4
      GO TO 390
  350 CALL STRFLT(3,3,5)
      LPOL=5
      GO TO 390
  360 DO 361 I=1,8
      AUTORO(I,1)=AUTORO(I,1)*1.467
  361 AUTORO(I,2)=AUTORO(I,2)*0.01667
C     DETERMINE FORWARD SPEED
      SPEED=SQRT((V(3,1)**2)+(V(3,2)**2))
C     DETERMINE PROPER RATE OF DESCENT FOR FORWARD SPEED
      DO 370 I=1,8
  370 IF (SPEED .GE. AUTORO(I,1)) VERCOM=AUTORO(I,2)
C     DETERMINE REQUIRED TOTAL VELOCITY
      REQVEL=SQRT((SPEED**2)+(VERCOM**2))
C     DETERMINE REQUIRED ANGLE OF DIVE
      REQANG=-ATAN(VERCOM/SPEED)
C     ADJUST THROTTLE TO OBTAIN PROPER SPEED
      IF (V(3,4) .LE. REQVEL) GO TO 373
      THROTL(3)=-((WEIGHT(3)*HDECEL/G)+DRAG(3)+WEIGHT(3)*
     1SIN(V(3,6)))/(THCON(3)*COS(ALPHA(3)))
      GO TO 374
  373 THROTL(3)=((WEIGHT(3)*HACCEL/G)+DRAG(3)+WEIGHT(3)*
     1SIN(V(3,6)))/(THCON(3)*COS(ALPHA(3)))
C     ADJUST DIVE ANGLE TO PROPER SETTING + OR - 1.5 DEGREES
  374 IF (V(3,6) .LT. (REQANG-1.5*RAD)) GO TO 375
      IF (V(3,6) .GT. (REQANG+1.5*RAD)) GO TO 376
      CALL STRFLT(3,1,4)
      LPOL=7
      GO TO 390
  375 CALL CLIMB1(3,1.1,1,4)
      LPOL=7
      GO TO 390
  376 CALL DIVE1(3,0.95,1,4)
      LPOL=7
      GO TO 390
  390 CONTINUE
C
C     STORE COORDINATES AT PRINTOUT TIMES
```

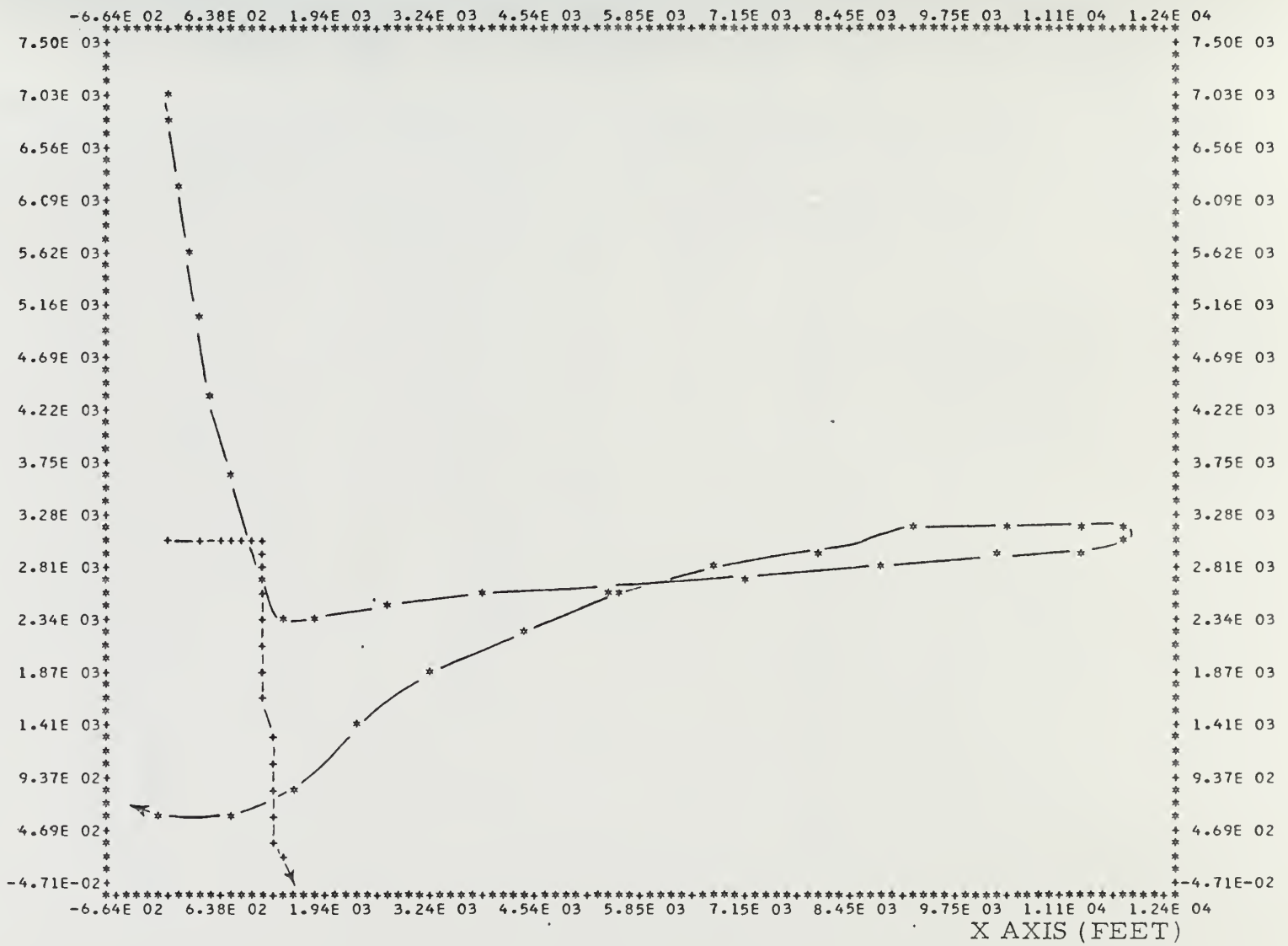
```
C
      IF (TIME.LT.TEST) GO TO 430
      KK3=KK1+61
      XXXX(KK1)=R(1,1)
      XXX(KK1,1)=R(1,1)
      XXXX(KK3)=R(3,1)
      XXX(KK3,2)=R(3,1)
      YYY(KK1)=R(1,2)
      YYY(KK3)=R(3,2)
      ZZZ(KK1,1)=R(1,3)
      ZZZ(KK3,2)=R(3,3)
      TEST=TEST+DTPO
      KK1=KK1+1
  430 CONTINUE
C
      RETURN
      END
```



PLOT OF FIGHTER(*) VS. HELICOPTER(+) IN THE Y-Z PLANE



PLOT OF FIGHTER (*) VS. HELICOPTER (+) IN THE X-Y PLANE



PLOT OF FIGHTER(*) VS. HELICOPTER(+) IN THE X-Z PLANE

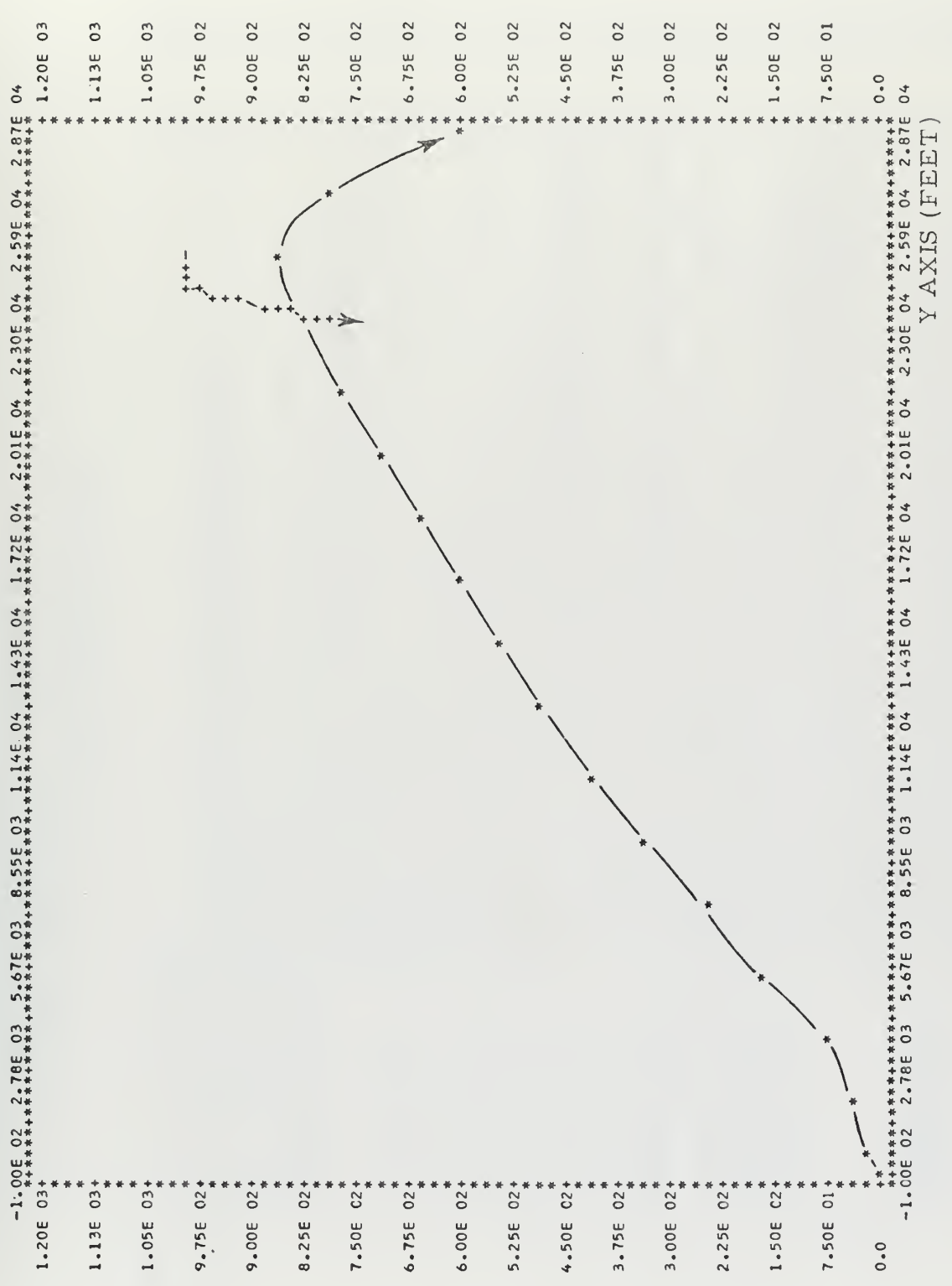
III. SAMPLE POLICY SUBROUTINE

MISSILE VS. HELICOPTER

```
C **** FIGHTER COMMANDS ****
C
      GO TO (110,120,130),JPOL
  110 CONTINUE
      CALL CAPFLT(1,2)
      GO TO 190
  120 CONTINUE
  130 CONTINUE
  190 CONTINUE
C
C **** MISSILE COMMANDS ****
C
      GO TO (210,220,230,240,250),KPOL
  210 CONTINUE
      CALL LAUNCH(2,0.0,3,5)
  220 CONTINUE
      IF ((TIME-TLAUN(2)) .GT. TBURN1) GO TO 230
      CALL DIVE1(2,0.0,1,4)
      KPOL=2
      GO TO 290
  230 CONTINUE
      IF ((TIME-TLAUN(2)) .GT. TGUIDE(2)) GO TO 240
      CALL DIVE1(2,0.0,1,5)
      KPOL=3
      GO TO 290
  240 CONTINUE
      ITAU(2)=1
      TAU(2,1)=0.2
  250 CONTINUE
      CALL PRONAV(2,1,5)
      KPOL=5
  290 CONTINUE
C
C **** HELICOPTER COMMANDS ****
C
      GO TO (310,320,370),LPOL
  310 IMISS=1
C     ACCELERATE TO 140 MPH IF POSSIBLE
      IF (V(3,4) .GE. 140.0*1.467) GO TO 320
      IF (RREL(3,4) .LE. 15000.) GO TO 325
      THFOR=WEIGHT(3)*HACCEL/G
      THROTL(3)=THFOR/THCON(3)
      CALL STRFLT(3,1,4)
      GO TO 390
  320 IF (RREL(3,4) .LE. 15000.) GO TO 325
      CALL STRFLT(3,3,5)
      LPOL=2
      GO TO 390
C     CONVERT INITIAL DATA TO PROGRAM UNITS
  325 DO 361 I=1,8
      AUTORO(I,1)=AUTORO(I,1)*1.467
  361 AUTORO(I,2)=AUTORO(I,2)*0.01667
C     DETERMINE FORWARD SPEED
      SPEED=SQRT((V(3,1)**2)+(V(3,2)**2))
C     DETERMINE PROPER RATE OF DESCENT FOR FORWARD SPEED
      DO 370 I=1,8
  370 IF (SPEED .GE. AUTORO(I,1)) VERCOM=AUTORO(I,2)
C     DETERMINE REQUIRED TOTAL VELOCITY
      REQVEL=SQRT((SPEED**2)+(VERCOM**2))
C     DETERMINE REQUIRED ANGLE OF DIVE
      REQANG=-ATAN(VERCOM/SPEED)
C     ADJUST THROTTLE TO OBTAIN PROPER SPEED
      IF (V(3,4) .LE. REQVEL) GO TO 373
      THROTL(3)=-((WEIGHT(3)*HDECEL/G)+DRAG(3)+WEIGHT(3)*
     1SIN(V(3,6)))/(THCON(3)*COS(ALPHA(3)))
```

```
      GO TO 374
  373 THROTL(3)=((WEIGHT(3)*HACCEL/G)+DRAG(3)+WEIGHT(3)*
     1SIN(V(3,6)))/(THCON(3)*COS(ALPHA(3)))
C     ADJUST DIVE ANGLE TO PROPER SETTING + OR - 1.5 DEGREES
  374 IF (V(3,6) .LT. (REQANG-1.5*RAD)) GO TO 375
      IF (V(3,6) .GT. (REQANG+1.5*RAD)) GO TO 376
      CALL STRFLT(3,1,4)
      LPOL=3
      GO TO 390
  375 CALL CLIMB1(3,1.1,1,4)
      LPOL=3
      GO TO 390
  376 CALL DIVE1(3,0.95,1,4)
      LPOL=3
      GO TO 390
  390 CONTINUE
```

IV. COMPUTER PLOT OF MISSILE VS. HELICOPTER



PLOT OF MISSILE(*) VS. HELICOPTER (+) IN THE Y-Z PLANE

APPENDIX C

I. MAIN PROGRAM LISTING

```
CMAIN                                                                   C     001
C     ORDERS OPERATIONS                                                 C     002
C                                                                       C     003
C ***********************************************************************C     004
      REAL MACHCL,MACHCD,MACHT1,MACHT2,MACHTA,MACHLM,MACHMX             C     005
      REAL LAMDA0,LAMBDA,LIFT,MACH,IMPLSE,KLAUN,MINMR                   C     006
      REAL LAT,LONG,LAT0,LONG0                                          C     007
      COMMON/CATMOS/ TEMP(3),PRES(3),DENS(3),SOUND(3),MACH(3),Q(3)      C     008
      COMMON/VECTOR/ R(3,6),RREL(6,6),V(3,6),VREL(6,6)                  C     009
      COMMON/AERO/   THRUST(3),ALPHA(3),CODRAG(3),COLIFT(3),DRAG(3),    C     010
     1               LIFT(3),WEIGHT(3),AREA(3),ASMAX(3),CLMAX(3),WO(3), C     011
     2               CWDOT(3),THROTL(3),BETA(3)                         C     012
      COMMON/RATE/   OMEGA(3,4),OMEGAR(6,4),OMEGAB(6,4),RDOT(6),VDOT(3),C     013
     1               THDOT(3),PHIDOT(3)                                 C     014
      COMMON/ATT/    AZMUTH(6),ELEV(6),ROLL(3),ROLLR8(3),BEARNG(6)      C     015
     1               ,ELEVMX(3),AZMAX(3),BANK(3)                        C     016
      COMMON/ACCEL/  ACOM(3),ACOMA(3),ACOMD(3),AOUT(3),AOUTA(3),AOUTD(3)C     017
     1               ,GFORCE(3),ACLMAX(3),AMAX(3),F(3)                  C     018
      COMMON/UNIT/   UNITV(3,3),UNITPV(3,3),UNITR(6,3),UNITL(3,3),      C     019
     1               UNITT(3,3),A(3,3),D(3,3),A1(3,3),D1(3,3)           C     020
      COMMON/INTEG/  HDP,T,VAR(18),DER(18),TEMPS(9,18),VARSTR(18),ICNT, C     021
     1               STIME,TIME,DTO,HMIN,HMX,HM,ERTEST,DTMIN,DT,SHDP,NH C     022
      COMMON/FLAG/   JPOL,KPOL,LPOL,MPOL,NPOL,JINTEG,ISTART,ILAUN,JATMOSC     023
     1               ,ISTORE,IMISS,KINTEG,ICAP                          C     024
      COMMON/PRINT/  IPRINT(20),TPO,TOTAL,DTPO,NPRINT,TITLE(18)         C     025
      COMMON/MISSL/  TGUIDE(3),TLAUN(3),TBURN1,TBURN2,KLAUN,RMTMAX,MINMRC     026
     1               ,ABOOST(3),IMPLSE(3),WBURN(3)                      C     027
      COMMON/ANALYT/ CDO(3),B(3),ALPHAO(3),CDOCON(3),BCON(3),THCON(3),   C     028
     1               TABCON(3),SLOPE(3)                                 C     029
      COMMON/CONST/  G,RAD,ITAU(3),TAU(3,5),DATA(200)                   C     030
      COMMON/NAVIG/  SGAMA(3),LAMDA0(3),LAMBDA(3),DVTH(3),DVPHI(3)      C     031
      COMMON/TABLE/  JVEH(3),MACHCL(3,42),CLMAXT(3,1,42),MACHCD(3,16),  C     032
     1               CLTCD(3,32),CDT(3,16,32),MACHT1(3,16),ALT1(3,16),  C     033
     2               THT1(3,16,16),MACHT2(3,16),ALT2(3,16),THT2(3,16,16)C     034
     3               ,CLTA(3,13),MACHTA(3,22),ALPHAT(3,22,13),WDOT(3),  C     035
     4               WDOT1(3,16,16),WDOT2(3,16,16),ALTM(3,14),MACHMX(3),C     036
     5               MACHLM(3,14)                                       C     037
      COMMON/EXTRA/  EXTR(20),IEXTRA(10)                                C     038
      COMMON/VEH/    ACTION(3,2),DASMX(3,5),DUMCL(3,5),ACTNO(3)         C     039
      COMMON/GEOCNT/ RO(3),RODOT(3),RRDOT(3,6),UNITRR(3,3),UNITPP(3,3), C     040
     1               UNITLL(3,3),FORCE(3,3),IROT8,RR(3,6),VI(3,6),      C     041
     2               RI(3,6),OMEGAE,LAT0,LONG0,ALT(3),LAT(3),LONG(3),   C     042
```

```
     3                GRANGE(3)                                                C     043
      COMMON/SAVREL/ SAVMIS(1,6)                                               C     044
      COMMON/PLOTT/    XXXX(122),XXX(122,2),YYY(122),ZZZ(122,2)
C     SC4060 IS IN COMMON WITH MAIN,INCOND, AND OUTPUT                         C     045
      COMMON/SC4060/I4060                                                      C     046
      DOUBLE PRECISION T,HDP,SHDP                                              C     047
C ***********************************************************************     C     048
C                                                                              C     049
C  SET ALL VALUES IN COMMON/PLOTT/ TO 9.0 E 05
      DO 30 I=1,122                                                                C
      XXXX(I)=9.0E05                                                             H
      YYY(I)=9.0E05                                                                A
      DO 30 J=1,2                                                                  N
      XXX(I,J)=9.0E05                                                              G
      ZZZ(I,J)=9.0E05                                                              E
30    CONTINUE
C  ZERO ALL VALUES IN COMMON/TABLE/--OTHER COMMON VALUES SET IN                C     050
C                                    SUBROUTINE INCOND                         C     051
      EQUIVALENCE(ZZERO(1),MACHCL(1,1))                                        C     052
      EQUIVALENCE (XZERO(1),RO(1))                                             C     053
      DIMENSION ZZERO(1)                                                       C     054
      DIMENSION XZERO(1)                                                       C     055
C  SET LAST VARIABLE IN COMMON TO ARBITRARY VALUE                              C     056
      MACHLM(3,14)=1.0                                                         C     057
C  COMMON BLOCK CONTAINS 6500 VARIABLES--0.233 SECS TO ZERO 7044 O.S.          C     058
      DO 5 I=1,7000                                                            C     059
      IF(MACHLM(3,14).EQ.0.0) GO TO 8                                          C     060
    5 ZZERO(I)=0.0                                                             C     061
    8 CONTINUE                                                                 C     062
      DO 100 I=1,200                                                           C     063
  100 DATA(I) = 0.0                                                            C     064
C READ FIRST DATA CARD GIVING NUMBER OF CASES IN RUN                           C     065
C
      INTEGER*4 INGAGE,OTGAGE                                                      C
      COMMON /GAGE/ INGAGE,OTGAGE                                                H
      WRITE (6,3)                                                                  A
    3 FORMAT (//' ENTER INGAGE AND OTGAGE')                                        N
      READ (5,2) INGAGE,OTGAGE                                                     G
    2 FORMAT (2I2)                                                                 E
      READ (INGAGE,1) MAXKAS
    1 FORMAT(I2)                                                               C     067
      KASE=1                                                                   C     068
  110 ISTART = 0                                                               C     069
      CALL TABLER                                                              C     070
C ZERO ALL VALUES IN COMMON GEOCNT                                             C     071
      LCNGO=1.0                                                                C     072
      DO 115 I = 1,125                                                         C     073
      IF(LCNGO.EQ.0.0) GO TO 118                                               C     074
```

```
  115 XZERO(I)=0.0                                                      C     075
  118 CONTINUE                                                          C     076
      CALL INCOND                                                       C     077
      CALL POLICY                                                       C     078
      CALL INITS(1)                                                     C     079
  120 CONTINUE                                                          C     080
      CALL POLICY                                                       C     081
C                                                                       C     082
C **** TO STORE VALUES AT LAUNCH ****                                   C     083
      IF (ISTORE .NE. 1) GO TO 130                                      C     084
      CALL STORE                                                        C     085
  130 CONTINUE                                                          C     086
C                                                                       C     087
C **** BACKUP FOR FIXED APPROACH TO LAUNCH ****                         C     088
      IF (JINTEG .EQ. 1 .OR. JINTEG .EQ. 2) GO TO 200                   C     089
      IF (ILAUN .NE. 2 ) GO TO 200                                      C     090
      CALL INITS(3)                                                     C     091
      CALL POLICY                                                       C     092
      CALL DAUX                                                         C     093
  200 CONTINUE                                                          C     094
C                                                                       C     095
      CALL AUXCOM                                                       C     096
      IF (TIME  .GE. TOTAL) GO TO 180                                   C     097
      CALL INTGRT                                                       C     098
C                                                                       C     099
  170 CONTINUE                                                          C     100
      IF (ISTART .EQ. 0) ISTART=1                                       C     101
      GO TO 120                                                         C     102
C                                                                       C     103
C
  180 CALL OSPLOT(YYY,XXX,122,122,2,0,0,0,0,0)
      WRITE(OTGAGE,9991)
      CALL OSPLOT(YYY,ZZZ,122,122,2,0,0,0,0,0)                                C
      WRITE(OTGAGE,9992)                                                        H
      CALL OSPLOT(XXXX,ZZZ,122,122,2,0,0,0,0,0)                                   A
      WRITE(OTGAGE,9993)                                                            N
 9991 FORMAT(' ',99X,'Y AXIS (FEET)')                                                 G
 9992 FORMAT(' ',99X,'Y AXIS (FEET)')                                                  E
 9993 FORMAT(' ',99X,'X AXIS (FEET)')
C
      IF(I4060.EQ.0)GO TO 190                                           C     104
      IF (KASE.EQ.MAXKAS) END FILE 11                                   C     105
  190 IF(KASE.GE.MAXKAS)CALL EXIT                                       C     106
      KASE=KASE+1                                                       C     107
      GO TO 110                                                         C     108
      END                                                               C     109
```

```
C                                                                      OSPL0000
C-----------------------------------------------------------------     OSPL0010
C                                                                      OSPL0020
C    A. IDENTIFICATION:                                                OSPL0030
C       TITLE:              GENERAL PURPOSE PLOTTING                   OSPL0040
C       SUBROUTINE NAME:    OSPLOT (J5-NPGS-OSPLOT)                    OSPL0050
C       DATE:               IMPLEMENTED IN FEB., 1969                  OSPL0060
C       PROGRAMMER:         ROY JOHNSON                                OSPL0070
C                                                                      OSPL0080
C                                                                      OSPL0090
C    B. PURPOSE:                                                       OSPL0100
C       THIS SUBROUTINE PLOTS ANY NUMBER OF SETS OF POINTS ON A        OSPL0110
C       SINGLE GRAPH, OUTPUTTING ON THE PRINTER. THE DIFFERENT         OSPL0120
C       POINT SETS ARE REPRESENTED BY DIFFERENT PRINT CHARACTERS,      OSPL0130
C       PROVISION IS MADE FOR AUTOMATIC OR MANUAL SCALING.             OSPL0140
C                                                                      OSPL0150
C                                                                      OSPL0160
C    C. USAGE:                                                         OSPL0170
C       1. CALLING ARGUMENTS:  CALL OSPLOT(X,Y,N,NDIM,NCUR,ISCALE,     OSPL0180
C                                          XL,XH,YL,YH)                OSPL0190
C          X(I)   - ARRAY OF X AXIS COORDINATES DIMENSIONED NDIM IN    OSPL0200
C                   THE CALLING PROGRAM.                               OSPL0210
C                                                                      OSPL0220
C          Y(I,J)- ARRAY OF NCUR Y AXIS VARIABLES EACH OF LENGTH N.    OSPL0230
C                  THE FIRST VARIABLE IS STORED IN Y(I,1), THE         OSPL0240
C                  SECOND VARIABLE IN Y(I,2), ETC.  DIMENSION OF Y     OSPL0250
C                  IS Y(NDIM,NCUR) IN THE CALLING PROGRAM.             OSPL0260
C                                                                      OSPL0270
C          N      - NUMBER OF POINTS TO BE PLOTTED FOR EACH GRAPH.     OSPL0280
C                                                                      OSPL0290
C          ISCALE- SCALE CONTROL VARIABLE.  IF ISCALE=0 THE PLOTS      OSPL0300
C                  WILL BE AUTOSCALED ON THE INPUT VARIABLES.  IF      OSPL0310
C                  ISCALE=1 THE SCALE IS SET IN X BY XL AND XH AND     OSPL0320
C                  IN Y BY YL AND YH.                                  OSPL0330
C                                                                      OSPL0340
C       2. ADVANTAGES:                                                 OSPL0350
C          BY PLOTTING ONE LINE AT A TIME ONLY 101 BYTES ARE NEEDED    OSPL0360
C          TO STORE GRID INFORMATION.                                  OSPL0370
C                                                                      OSPL0380
C       3. DISADVANTAGES:                                              OSPL0390
C          ALL CURVES MUST BE SUPPLIED AT ONCE, WHICH MEANS MORE       OSPL0400
C          STORAGE IS REQUIRED IN THE CALLING PROGRAM.                 OSPL0410
C                                                                      OSPL0420
C                                                                      OSPL0430
C-----------------------------------------------------------------     OSPL0440
```

```
C                                                                       OSPL0450
C                                                                       OSPL0460

      SUBROUTINE OSPLOT(X,Y,N,NDIM,NCUR,ISCALE,XL,XH,YL,YH)             OSPL0470
C*MULTIPLE CURVE PRINTER PLOT ROUTINE                                   OSPL0480
C  PARAMETERS: X(I)=ARRAY OF X AXIS COORDINATES DIMENSIONED NDIM IN     OSPL0490
C                   THE CALLING PROGRAM.                                OSPL0500
C              Y(I,J)=ARRAY OF NCUR Y AXIS VARIABLES EACH OF LENGTH N.  OSPL0510
C                     THE FIRST VARIABLE IS STORED IN Y(I,1), THE       OSPL0520
C                     SECOND VARIABLE IN Y(I,2),ETC. DIMENSION OF Y IS  OSPL0530
C                     Y(NDIM,NCUR) IN THE CALLING PROGRAM.              OSPL0540
C              N=NUMBER OF POINTS TO BE PLOTTED FOR EACH GRAPH.         OSPL0550
C              ISCALE=SCALE CONTROL VARIABLE. IF ISCALE=0 THE PLOTS WILLOSPL0560
C                     BE AUTOSCALED ON THE INPUT VARIABLES. IF ISCALE=1 OSPL0570
C                     THE SCALE IS SET IN X BY XMIN AND XMAX AND IN Y BYOSPL0580
C                     YMIN AND YMAX.                                    OSPL0590
      DIMENSION XS(11),YS(17),X(1),Y(NDIM,NCUR)                         OSPL0600
      INTEGER *2 IGRID(101)                                             OSPL0610
      INTEGER *4 ICHAR(6)/'  * ',' + ',' = ',' = ',' . ','   '/         OSPL0620
      IF(ISCALE.EQ.1)GO TO 32                                           OSPL0630
C                                                                       OSPL0640
C  AUTOSCALING IF SO DESIRED.                                           OSPL0650
      XMAX=-1.0E 20                                                     OSPL0660
      XMIN=-XMAX                                                        OSPL0670
      YMAX=XMAX                                                         OSPL0680
      YMIN=-XMAX                                                        OSPL0690
      DO 25 I=1,N                                                       OSPL0700
      IF (X(I) .GE. 8.0E05) GO TO 25                                    ADDITION
      IF(X(I).GT.XMAX)XMAX=X(I)                                         OSPL0710
      IF(X(I).LT.XMIN)XMIN=X(I)                                         OSPL0720
   25 CONTINUE                                                          OSPL0730
      DO 31 J=1,NCUR                                                    OSPL0740
      DO 31 I=1,N                                                       OSPL0750
      IF (Y(I,J).GT.8.0E05) GO TO 31                                    ADDITION
      IF(Y(I,J).GT.YMAX)YMAX=Y(I,J)                                     OSPL0760
      IF(Y(I,J).LT.YMIN)YMIN=Y(I,J)                                     OSPL0770
   31 CONTINUE                                                          OSPL0780
      GO TO 34                                                          OSPL0790
C                                                                       OSPL0800
C  AUTOSCALING NOT DESIRED, USE VALUES PROVIDED                         OSPL0810
   32 XMIN=XL                                                           OSPL0820
      XMAX=XH                                                           OSPL0830
      YMIN=YL                                                           OSPL0840
      YMAX=YH                                                           OSPL0850
C                                                                       OSPL0860
C  GET RANGES.                                                          OSPL0870
   34 XR=XMAX-XMIN                                                      OSPL0880
```

```
      IF(XR.EQ.0.0)XR=1.0E-20                                          OSPL0890
      YR=YMAX-YMIN                                                     OSPL0900
      IF(YR.EQ.0.0)YR=1.0E-20                                          OSPL0910
C                                                                      OSPL0920
C  DETERMINE WHETHER OR NOT BOTH POSITIVE AND NEGATIVE VALUES OCCUR IN OSPL0930
C  REPECTIVELY X AND Y.  IF THIS OCCURS, PLACE AXIS APPROPRIATELY.     OSPL0940
      XT=XMAX*XMIN                                                     OSPL0950
      YT=YMIN*YMAX                                                     OSPL0960
      IF(XT.LT.0.0)IYAX=100.0*(-XMIN)/XR+1.5                           OSPL0970
      IF(YT.LT.0.0)IXAX=64.0*YMAX/YR+1.5                               OSPL0980
C                                                                      OSPL0990
C  GET GRID ELEMENT DIMENSIONS.                                        OSPL1000
      XINCR=XR/10.0                                                    OSPL1010
      YINCR=YR/16.0                                                    OSPL1020
C                                                                      OSPL1030
C  STORE LABELS.                                                       OSPL1040
      XS(1)=XMIN                                                       OSPL1050
      YS(1)=YMAX                                                       OSPL1060
      DO 46 I=2,11                                                     OSPL1070
   46 XS(I)=XS(I-1)+XINCR                                              OSPL1080
      DO 47 I=2,17                                                     OSPL1090
   47 YS(I)=YS(I-1)-YINCR                                              OSPL1100
C                                                                      OSPL1110
C  WRITE X LABELS.                                                     OSPL1120
      WRITE(6,10)(XS(I),I=1,11)                                        OSPL1130
      II=1                                                             OSPL1140
      KK=0                                                             OSPL1150
      DO 146 LINE=1,65                                                 OSPL1160
C                                                                      OSPL1170
C  SET LINE TO BLANKS.                                                 OSPL1180
      DO 101 J=1,101                                                   OSPL1190
  101 IGRID(J)=ICHAR(6)                                                OSPL1200
C                                                                      OSPL1310
C  FIND Y VALUES ON THE LINE AND PLACE THE CHARACTER CORRESPONDING TO  OSPL1320
C  THE APPROPRIATE POINT SET IN THAT X VALUE LOCATION IN LINE.         OSPL1330
      DO 125 J=1,NCUR                                                  OSPL1340
      DO 125 I=1,N                                                     OSPL1350
      IF (Y(I,J).GT.8.0E05) GO TO 125                                  ADDITION
      IPTY=64.0*(YMAX-Y(I,J))/YR+1.5                                   OSPL1360
      IF(IPTY.GT.65)IPTY=65                                            OSPL1370
      IF(IPTY.LT.1)IPTY=1                                              OSPL1380
      IF(IPTY-LINE)125,117,125                                         OSPL1390
  117 IPTX=100.0*(X(I)-XMIN)/XR+1.5                                    OSPL1400
      JC=MOD(J,4)                                                      OSPL1410
      IF(JC)119,118,119                                                OSPL1420
  118 IGRID(IPTX)=ICHAR(4)                                             OSPL1430
      GO TO 125                                                        OSPL1440
  119 IF (IGRID(IPTX).EQ.ICHAR(1)) IGRID(IPTX)=ICHAR(3)                ADDITION
```

```
      IF (IGRID(IPTX).NE.ICHAR(1)) IGRID(IPTX)=ICHAR(JC)                ADDITION
  125 CONTINUE                                                          OSPL1460
C                                                                       OSPL1470
C   PRINT LINE WITH OR WITHOUT LABELS, DEPENDING ON WHETHER OR NOT THEY OSPL1480
C   BELONG THERE.                                                       OSPL1490
      IF(KK)133,133,134                                                 OSPL1500
  133 WRITE(6,20)YS(II),(IGRID(I),I=1,101),YS(II)                       OSPL1510
      II=II+1                                                           OSPL1520
      GO TO 135                                                         OSPL1530
  134 WRITE(6,30)(IGRID(I),I=1,101)                                     OSPL1540
  135 KK=KK+1                                                           OSPL1550
      IF(KK-4)146,136,146                                               OSPL1560
  136 KK=0                                                              OSPL1570
  146 CONTINUE                                                          OSPL1580
C                                                                       OSPL1590
C   WRITE X LABELS.                                                     OSPL1600
      WRITE(6,40)(XS(I),I=1,11)                                         OSPL1610
      RETURN                                                            OSPL1620
   10 FORMAT(1H1,1PE15.2,10E10.2/10X,1H*,20(5H+****),2H+*)              OSPL1630
   20 FORMAT(1PE10.2,1H+,101A1,1H+,E9.2)                                OSPL1640
   30 FORMAT(10X,1H*,101A1,1H*)                                         OSPL1650
   40 FORMAT(10X,1H*,20(5H+****),2H+*/1PE16.2,10E10.2)                  OSPL1660
      END                                                               OSPL1670
```

LIST OF REFERENCES

1. The RAND Corporation Memorandum RM-5759-PR, TACTICS: A Three-Body, Three-Dimensional Intercept Simulation Program, by J. H. Hutcheson and R. L. Segerblom, October 1969.

2. Department of the Army Technical Manual, TM55-1520-221-10, Operator's Manual Army Model AH-1G Helicopter, April 1969.

3. NAVTRA DEVCEN Technical Report 1205-1, Simulation of Helicopter and V/STOL Aircraft, by J. R. Toler, W. McIntyre, and M. P. Coffee, September 1963.

4. Stepniewski, W. Z., Introduction to Helicopter Aerodynamics, v. 1, Rotorcraft Publishing Committee, 1950.

5. Harris, R. J., Sloan, L. H., and Ulrich, K. W., Typical Helicopter Performance Calculation, Rotocraft Publishing Committee, 1952.

6. Gage, W. R., A Study of the Use of RAND's TACTICS Program Applied to Simulating the Flight of a Helicopter in a Tactical Situation, M. S. Thesis, Naval Postgraduate School, Monterey, California 1970.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department of Operations Analysis (Code 55) Naval Postgraduate School Monterey, California 93940	1
4.	Asst. Professor G. E. Heidorn, Code 55Hd Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	3
5.	Lieutenant Robert Alex Maier, USN 1165 Lazy Lake Road East Dunedin, Florida 33528	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3 REPORT TITLE

A Method for the Simulation of a Helicopter in a Tactical Environment Using RAND's TACTICS Program

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

Master's Thesis; March 1971

5 AUTHOR(S) (First name, middle initial, last name)

Robert Alex Maier

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1971	69	6

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

The TACTICS computer program developed by the RAND Corporation has the capability of simulating three-body, three-dimensional tactical situations. This study investigates the problem of using the TACTICS program when one of the bodies is to have the characteristics of a helicopter. The differences between the forces acting on a helicopter and the forces assumed by the TACTICS program are discussed, and a method to simulate the helicopter is presented. The basic idea of this method is to "fly" a hypothetical fixed-wing aircraft with parameters which enable it to perform like a helicopter. The programing required to simulate several basic helicopter maneuvers is discussed, and two examples of a helicopter in a tactical environment are presented.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Helicopter Flight Simulation						

Thesis 126215
M2772 Maier
c.1
A method for the simulation of a helicopter in a tactical environment using Rand's tactics program.